DataGeneral

Customer Documentation

Using CLASP (Class Assignment and Scheduling Package)

Using CLASP

(Class Assignment and Scheduling Package)

093-000422-00

For the latest enhancements, cautions, documentation changes, and other information about this product, please see the Release Notice (085-series) supplied with the software.

Ordering number 093-000422

Printed in the United States of America
Revision 00, March 1986

Using CLASP (Class Assignment and Scheduling Package)

093-000422

Revision History: Effective with:

Original Release - March 1986 AOS/VS Revision 7.00

Preface

Data General's CLASP, Class Assignment and Scheduling Package, can help you tailor process scheduling in AOS/VS for the specific needs of your computer site.

CLASP lets you create, monitor, and manage process classes and logical processors easily and quickly. Classes and logical processors can be useful on any MV/Family system, regardless of the number of job processors (physical processors, CPUs) it has.

Classes and logical processors aren't required to run an MV/Family computer that has multiple job processors -- AOS/VS can handle all job processors efficiently without them. However, if you need to tailor AOS/VS for your site's processing needs, CLASP will enable you to improve the performance of specific processes.

This manual tells how to use classes and logical processors. Its primary focus is CLASP, but it shows the use of other programs that pertain to classes and logical processors. The manual is organized as follows.

Chapter 1	introduces the concept of classes and logical processors -- including class scheduling and its advantages. Read this for background before using CLASP. It will help you understand the issues and plan your class environment. Chapter 1 also lists CLASP operating requirements and switches.
Chapter 2	describes using CLASP. It explains each CLASP menu entry and why you might want to use the entry; then it gives examples. This chapter is the heart of the manual. You may use it often -- long after you're familiar with classes.
Chapter 3	is an example of class use, from planning to implementation, at a sample computer site. Read it when you want some perspective on classes and their place in routine data processing.
Chapter 4	explains error conditions related to classes and logical processors.
Glossary	defines pertinent terms, like class and job processor. For any term you want defined, use the Glossary.

To create a system that uses classes fruitfully, you'll also need several programs supplied with AOS/VS. They are

- User Profile Editor, PREDITOR and/or
- Selective Preamble Editor, SPRED

Settings in user profiles and program preambles help determine the class in which a process will run. If your system uses only the default profiles and preambles, all processes will run in the same class, which means that class scheduling will have no effect.

- Process Environment Display, PED.

  PED can display class information on running processes.

CLI commands related to the use of classes and logical processors are LOCALITY, and PROCESS with /LOCALITY= switches. If your computer includes two or more job processors, the JPINITIALIZE command is needed to bring additional job processors on line.

Who Are You?

This manual assumes that you're an experienced and highly motivated data processing professional.

Classes and logical processors -- and CLASP -- involve nonstandard process scheduling. They're useful when you have one or more sets of processes that would benefit from nonstandard scheduling.

To use CLASP productively, you need some background in operating systems, specific knowledge of standard AOS/VS scheduling, and an understanding of the processes that run on your system. You must know enough about standard scheduling and your site's processes to recognize the need for nonstandard scheduling.

Also, you must be motivated to learn about classes, processes, and CLASP, and to create, test, and refine your definitions until you're satisfied with the results.

What Other Documentation Will You Need?

For material on other AOS/VS programs and on scheduling, you'll need parts of

- How to Generate and Run AOS/VS, 093-000243, for details on running PREDITOR, SPRED, and PED.

If you have the AOS/VS Performance Package, you'll want the appropriate templates and the

- AOS/VS Performance Package User's Manual, 093-000364.

Reader Please Note:

We use these conventions for command formats in this manual:

COMMAND required [optional] ... <nl>

Where	Means
COMMAND	You must type the command or argument as shown.
required	You must type an argument (like a filename).
[optional]	You have the option of entering this argument. Don't type the brackets; they only set off what's optional.
...	You may repeat the preceding entry or entries.
<nl>	Press the NEW LINE key on your terminal's keyboard. If there is no NEW LINE key, press the carriage return (RETURN) key.

All numbers are decimal, unless a number is noted as octal.

The book shows CLI commands in UPPERCASE; but you can type them in lowercase, uppercase, or any combination.

In example dialogs, we use

UNDERLINED CHARACTERS TO SHOW YOUR ENTRY

Contacting Data General

- If you have comments on this manual, please use the prepaid User Documentation Remarks Form that appears after the Index. We want to know what you like and dislike about the manual.

- If you need more manuals, please use the enclosed TIPS Order Form (USA only) or contact your local Data General sales representative.

End of Preface

Contents
========

Illustrations

Figure

Tables

Table

Chapter 1

Classes, Logical Processors, and CLASP

This chapter offers background on classes, logical processors, and CLASP (Class Assignment and Scheduling Package). The chapter will help you understand, plan, implement, and use classes and logical processors. Read it before using CLASP for the first time, and any time you want a review.

Major sections are

- About Classes and System Performance
- What Are Classes?
- What Are Logical Processors?
- Scheduling: Standard and Class
- About CLASP
- Benefits of Class Scheduling
- Error Conditions

About Classes and System Performance

CLASP eases the job of creating classes and logical processors, allowing you to tailor AOS/VS scheduling for specific needs. It has a monitor to give feedback on classes and logical processors -- providing a base for your class and processor definitions.

From the standpoint of performance (system throughput), there are three main components. They are

- Job processor (CPU) usage;
- Memory usage and memory contention issues; and
- Disk usage (disk I/O).

CLASP can help with the first component, job processor usage. For information on memory issues, you can run the optional AOS/VS Performance Package. For disk usage information, you can use the AOS/VS utility DISCO -- or disk monitoring tools included in the Performance Package.

Program timings and system response are the final measures of performance. Ultimately, they will tell you whether or not you've improved things.

What Are Classes?

A class is a set of processes for which you want special scheduling treatment. Usually, this treatment involves allotting a percentage of processor time. Each class defines at least one user and program locality (called a locality pair). A process will run in a specific class if its user and program localities match a locality pair defined for the class.

User locality is defined with PREDITOR in a user's profile. If the profile allows, a user can change user locality with CLI commands. Program locality is defined with the Selective Preamble Editor (SPRED) in a program file. The locality bounds (pairs) are defined with CLASP. User and program localities each range from 0 through 15. All localities (user and program localities 0 though 15), and therefore all processes, are included in the default class, DEFAULT.CLASS, that ships with AOS/VS.

For example, say you want a class for privileged users. You want every process with user locality 1 run in this class. You create the class via CLASP using a locality table (called a class matrix) as shown in Figure 1-1.

		Program Locality															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
U	0																
s	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
e	2																
r	3																
	4																
L	5																
o	6																
c	7																
a	8																
l	9																
i	10																
t	11																
y	12																
	13																
	14																
	15																

DEFAULT.CLASS
PRIVILEGED.USERS

Figure 1-1. A Class for Privileged Users

After creating the PRIVILEGED.USERS class, you allot it a specific amount of processor time, say 40%.

A user process can join this class by assuming a user locality of 1. (The program locality doesn't matter, since the class is defined in all 16 program localities. The determining factor is the user, not the program.) To start a user's process in the privileged class, run PREDITOR on user's profile and specify a default user locality of 1. Or, specify the user locality that puts the user in the default class, and allow the user to join the privileged class by changing locality to 1.

What's a Logical Processor?

A logical processor is a scheduling arrangement that -- usually -- includes a set of classes. It's active only while a job processor (physical processor) is connected to it. A logical processor named DEFAULT.LP is shipped with AOS/VS. At AOS/VS startup, the default job processor is connected to DEFAULT.LP.

If your computer has multiple job processors, you can add other job processors to DEFAULT.LP with the CLI command JPINITIALIZE. Still, there remains but one logical processor, DEFAULT.LP, running processes using standard AOS/VS scheduling.

CLASP lets you create additional logical processors (up to a total of 16), allot processor time to classes, and move a job processor from one logical processor to another logical processor. You can activate different logical processors for different environments -- perhaps based on the time of day.

For example, one application of class scheduling using two logical processors -- and only one job processor -- appears in Figure 1-2.

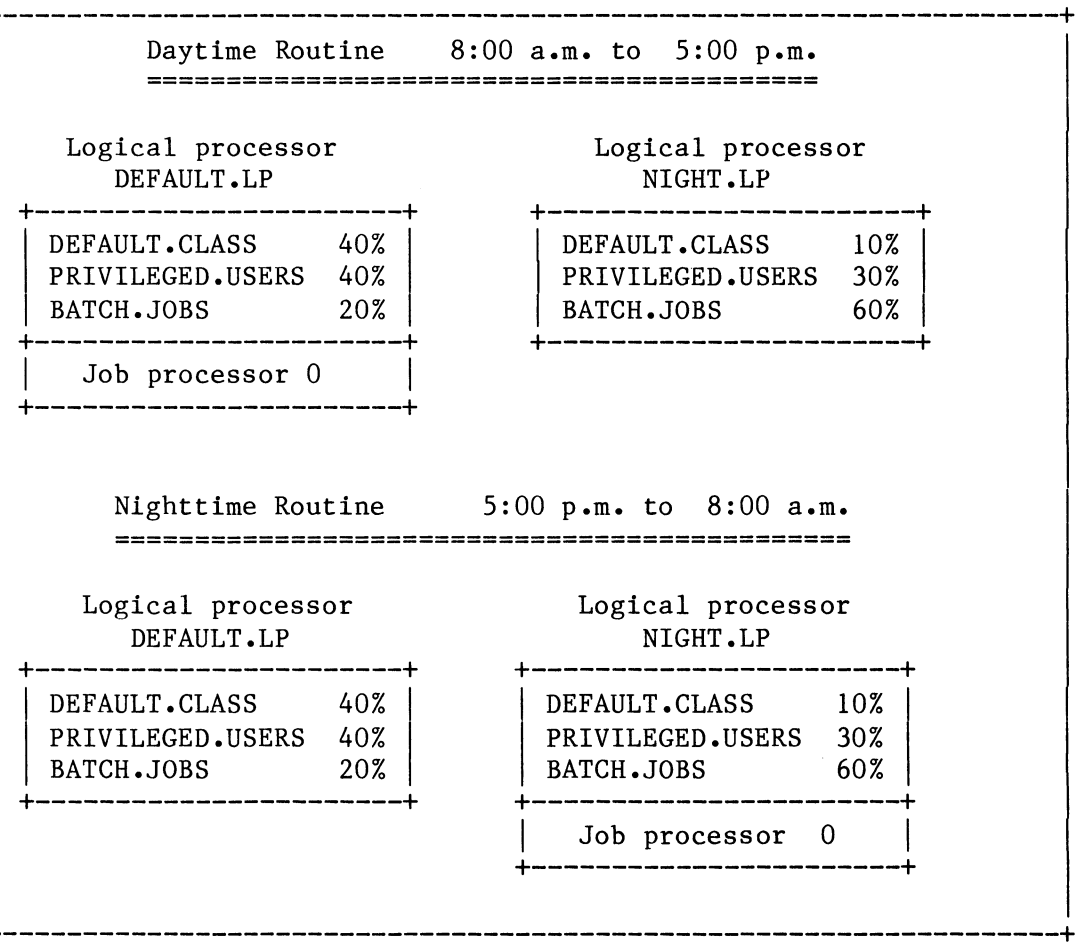

Figure 1-2. Using Different Logical Processors with One Job Processor

Figure 1-2 shows a logical processor that favors the default and privileged classes (interactive users) during the working day, and another logical processor that favors batch jobs at night. Moving the job processor from logical processor DEFAULT.LP to NIGHT.LP is a simple step with CLASP -- it can be done from the CLI via a CLASP script file.

If your computer has multiple job processors, there's even more flexibility. An application similar to Figure 1-2, but with two job processors, follows in Figure 1-3.

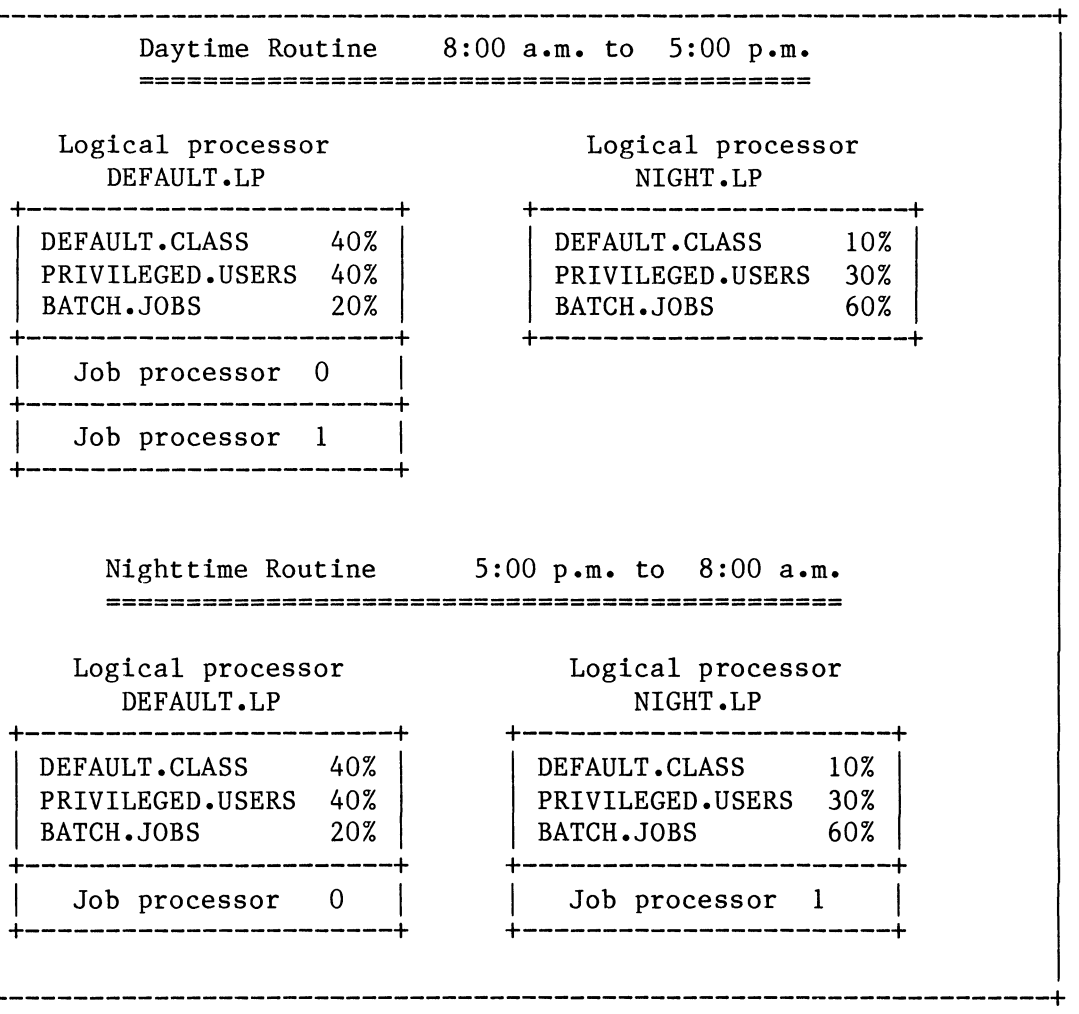

Figure 1-3. Using Different Logical Processors with Two Job Processors

Figure 1-3 shows two job processors connected to logical processor DEFAULT.LP during the day -- providing maximum processing power for interactive users during the day. At night, one job processor is moved to logical processor NIGHT.LP, giving all processes in class BATCH.JOBS a larger percentage of processing power.

Ultimately, with two job processors, you can create precisely tailored processing arrangements. Figure 1-4, next, shows such an arrangement.

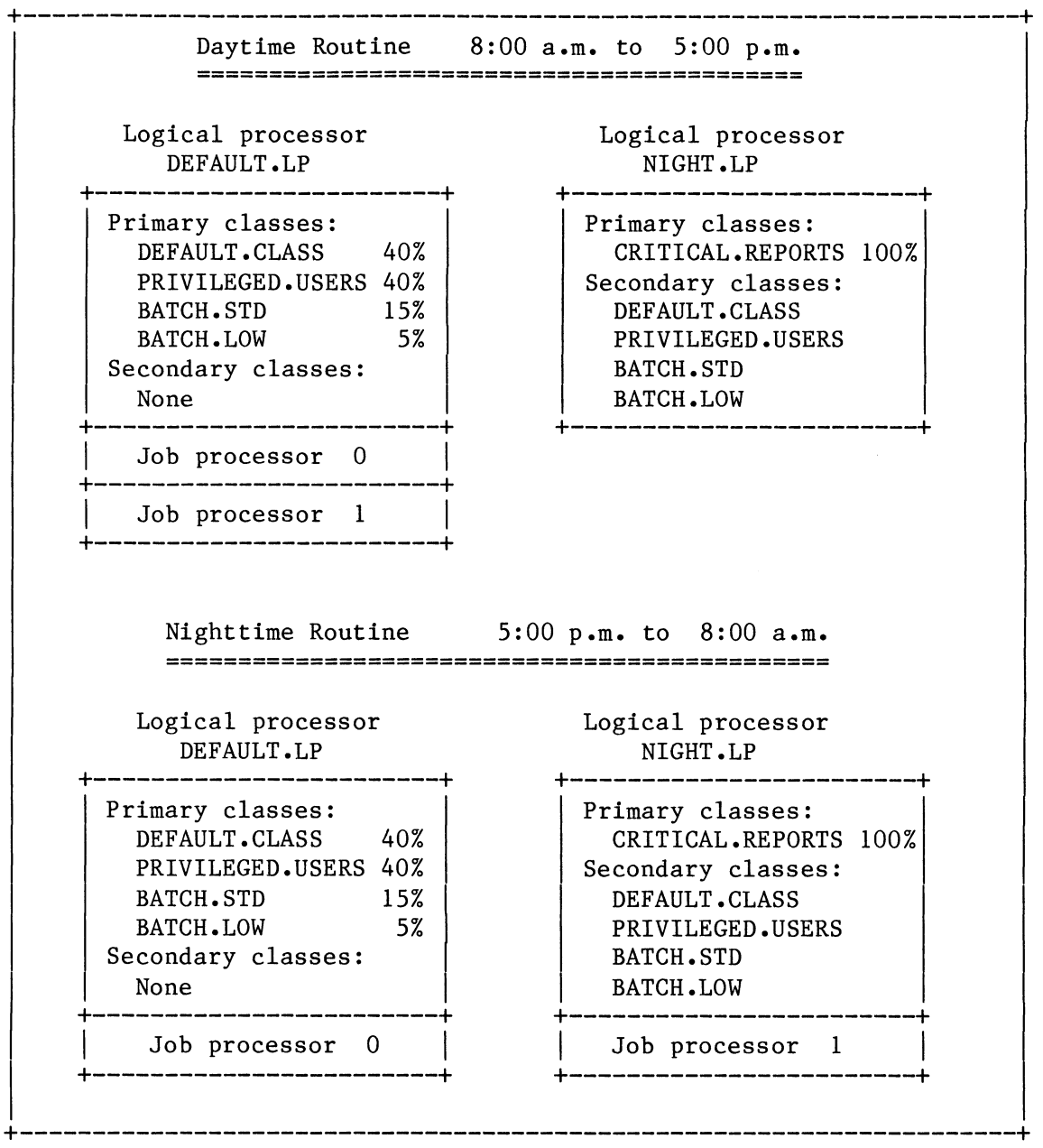

Figure 1-4. Tailored Arrangement, Two Logical Processors and Two Job Processors

Like the preceding figures, Figure 1-4 shows a logical processor that favors standard production during the day and a logical processor that produces a different environment at night. These processors include primary classes (with percentage amounts) and secondary classes (that get processing time only when no primary class wants it).

During the day, classes that run batch jobs get relatively little time, and class CRITICAL.REPORTS gets none, since CRITICAL.REPORTS's logical processor, NIGHT.LP, isn't connected to a job processor. At 5:00 p.m., a job processor is moved to NIGHT.LP and NIGHT.LP becomes active.

On NIGHT.LP, class CRITICAL.REPORTS is the only primary class. It's allotted 100%; processes in it get all the time they want. The other classes on NIGHT.LP are secondary classes on the same level; processes in these secondary classes compete for time not needed by CRITICAL.REPORTS. This arrangement allows the critical class all the time it needs on job processor 1 -- but when the report-creating processes have finished, processor 1 will share the load of other classes (like the DEFAULT.CLASS and PRIVILEGED.USERS) with processor 0.

Any of the processes in these classes can be started interactively (via XEQ or PROCESS) or started in a batch stream.

If someone tries to start a process in class CRITICAL.REPORTS before NIGHT.LP becomes active, the system will reject the PROCESS (or XEQ) command with an error message -- otherwise, the process would be stranded, unable to be scheduled.

Scheduling: Standard and Class

If class scheduling is disabled (default), or when processes that have time remaining within a set of classes compete, AOS/VS uses standard scheduling.

Standard Scheduling

Standard scheduling, which AOS/VS has used for years and which forms the basis for all scheduling, works as follows:

The AOS/VS scheduler runs on each job processor. When a job processor becomes free, that scheduler runs the highest priority, ready process.

The process may use the processor for a subslice period (32 milliseconds) if it wants.

At the end of a subslice; or if the process encounters a blocking event; or if a higher priority process becomes ready, AOS/VS reschedules. Again, the scheduler runs the highest priority, ready process. The chosen process may be the same process or a different one.

There are two methods of scheduling: round robin and heuristic. With round-robin scheduling, the scheduler tries to give each process at the same priority an equal amount of time. A process that uses a lot of processor time isn't penalized.

With heuristic scheduling, the system may reduce a process's internal priority, based on process behavior. A process that uses a lot of processor time is penalized -- relative to other, more interactive processes at the same priority.

The kind of scheduling a process gets depends on its group. The group is determined by process priority (range 0 to 511), with a few exceptions. Group definition rules and exceptions are detailed in How to Generate and Run AOS/VS. Generally, the following rules apply.

- Group 2 processes are scheduled heuristically (penalized for heavy processor demand). Within any group 2 priority level, the system favors interactive processes, in which people often pause for thought. Noninteractive (compute-bound) processes get reduced internal priority. Heuristic "favoring" often gives interactive processes better response time than they'd get under round-robin scheduling. By default, all user processes (and their sons) are group 2 processes.

- Group 1 and group 3 processes get round-robin scheduling. Compute-bound processes in these groups aren't penalized (although this may slow the response time of interactive processes at the same priority). A process's internal priority doesn't change, regardless of the amount of processor time it has consumed.

Standard scheduling is meant for general purpose systems. Since, by default, processes start in group 2, highly interactive timesharing processes are favored over compute-bound processes. But this interactive bias can be overcome -- as desired -- if you give user processes a priority that makes them group 1 or group 3. And, standard scheduling allows high priority processes (like the PMGR) to get all the time they need without penalty.

A bound processes of the same priority tend to get equal amounts of processor time, regardless of group. This works well in situations where you want to treat compute-bound processes equally.

However, standard scheduling, based on priority only, does have two limitations, as follows:

- You can't give one compute-bound process preference over another without starving the lower priority process.

  If you want to give one compute-bound process preference over others, and you give it higher priority, the process can consume all available processor time. Since AOS/VS gives control to the highest priority, ready process, when the process is ready, AOS/VS will give it control. If the process is always ready (as it would be if it were always computing and doing little I/O), then it will get all available processor time. AOS/VS will give it subslice after subslice, while lower priority processes get no time.

  For example, say two compute-bound processes are running at priority 5. Each process will get approximately 50% of the available processor time. If you change the priority of one process to 4 (higher priority), it will get about 100% of available time while the other gets no time. Standard scheduling provides no middle ground between even distribution (here, 50/50) and monopoly (here, 100/0).

- The second limitation of standard scheduling may appear when you give compute-bound processes a lower priority than interactive ones (you might do this to give users faster response time). If there are many interactive processes, they will consume all processing time: the lower priority, compute-bound processes would get no time. There's no way to give them even a tiny percentage of time. Thus compute-bound jobs might take hours or even days to complete.

For example, say a system has 60 timesharing users (which represent interactive processes) and 2 batch streams (which represent compute-bound processes). By default, the batch streams get a lower priority than the user processes. This gives the batch jobs less processing time, prolongs the turnaround time of each batch job, and increases the number of jobs stacked on the queue. Batch processing can occur at night -- but for some jobs, this isn't acceptable: daily turnaround is required. Standard scheduling offers no good solution to this problem.

With classes, you can specify the percentage of time for processes in a class. AOS/VS will then try to give that class the specified amount of processor time. This overcomes both limitations above.

As an additional benefit, you can force processes to run in a specific class, whether they are compute bound (batch) or interactive (user). Class scheduling extends your control over standard scheduling.

Class Scheduling

Part of the class definition procedure is to allot the class time on a logical processor. You can allot processor time to all 16 classes if you want. And, you can rank each class as follows:

Primary, with a percentage of processor time. Processes in any primary class can use up to their percentage allotment.

Secondary, with levels (up to 16 levels are allowed). After ready processes in primary classes get time, secondaries compete by level.

When class scheduling is enabled, the highest priority, ready process gets control as usual -- unless

- the process belongs to a primary class that's consumed its percentage and another primary-class process is ready; or

- the process belongs to a secondary class and a higher level secondary-class or a primary class process is ready.

In either of these cases, the process can't get time regardless of its priority.

For a primary class example, take the class PRIVILEGED.USERS shown in earlier figures. The processes in this class are user processes, so assume they're scheduled heuristically. Normally, heuristic scheduling penalizes processes that use a lot of processor time. But these user processes belong to a primary class allotted 40% of processor time. So, these users will be allowed to use up to 40% if they can possibly use it. They will get faster response time at the expense of other processes (but if these privileged processes can't use the time, it will be given to other primary- or secondary-class processes).

Or, say there are two compute-bound processes to run. You want one process to get twice as much time as the other. Without classes (the limitation described above), you couldn't do this. With classes,

you simply set up a primary class for each process (perhaps defined by program locality), then allot the preferred class twice the percentage of processor time as the other. For example -- depending on the importance of these processes -- you could give the preferred process class 75% and the other process class 25%.

The system enforces class percentages over a constant interval. The interval is the number of seconds the connected job processor(s) spends serving user processes (sum of CPU-seconds devoted to user processes). When the interval expires, the system zeros its figures (freeing any class that consumed its percentage) and starts counting again. The default interval is 4 seconds. So, in the 75/25 arrangement above, for 4 seconds AOS/VS would enforce a 75% limit on one class and a 25% limit on the other class. If one class reached its limit, its processes would become ineligible until the next interval.

The default interval of 4 seconds is a good general-purpose value. But if you have a class with interactive processes (like users in text editors) that tends to exhaust its percentage allotment, all the class's processes may appear blocked for a second or so during each interval. You can eliminate such delays in service by specifying a shorter interval -- say 1 or 2 seconds.

The system also zeros its figures (freeing classes that consumed their percentages) when the connected job processor(s) become idle. This allows primary-class processes whose class percentage is exhausted to run if no other primary-class process with time remaining wants time.

When the sum of all class percentage allotments equals 100%, the system will allow each class to use up to its percentage. Generally, we suggest that you maintain a sum of 100% -- not less, not more -- for primary classes on any logical processor. (A sum of less than 100% may result in needless scheduling overhead; a sum of more than 100% may result in some classes using all processor time, starving others.)

One exception to the 100% sum rule involves high priority server processes (like CEO , XODIAC , or INFOS II processes). If you create a class that includes these, the total percentage for that logical processor can exceed 100%. (For example, you might give the server class 100% and give other classes a total of 80%.) This works because these servers typically don't use a lot of processor time and thus will not use very much of the 100% -- but, if for some reason they need the time, they should be able to get it. You'd still need to give the server processes high priority, since you want them to get processor time first from among the classes that have time remaining.

Within primary classes whose percentage limit hasn't been reached, AOS/VS uses standard scheduling, as described above. AOS/VS also uses standard scheduling in logical processors for which class scheduling hasn't been enabled.

A process can run in only one job processor at a time. On a computer with more than one job processor, if your class percentages would force the system to leave a job processor idle, the system won't enforce your percentages. This can happen with a class that includes just one process, if you allot more than 50% of processor time to the class.

For example, say you have a computer with two job processors. It uses a logical processor with a class to which you've allotted 60%. Only one process runs in this class. Since only one processor at a time can run the process, the class can't get more than 50% of the time if both processors stay active. The class could get more than 50% only if AOS/VS left one processor idle, which would be inefficient. This topic is further explained in Chapter 2, "Monitor Logical Processors".

Generally, classes and logical processors are most useful when you want to deal with one or more exceptional processes. You can leave unexceptional processes alone, as part of the default class (name DEFAULT.CLASS) supplied with AOS/VS.

Figure 1-5, following, compares sample processes under standard scheduling (class scheduling disabled) and class scheduling (enabled).

Figure 1-5, left to right, shows four runs of the scheduler and the process selected on each run. The top portion applies with class scheduling disabled; the bottom portion with class scheduling enabled. In this logical processor, class A is allotted 50% of the processor's time. Class B and class X get 25% of its time.

In the top portion, the same process always has the highest priority and is always ready -- so it always gets control. No other process gets any time during the four scheduler runs.

In the bottom portion, with class scheduling enabled, the process in class A with priority 1 gets control twice. All the classes (A, B, and X) had time remaining. The priority 1 process ran because it had the highest priority. During scheduler runs 1 and 2, this process consumed class A's percentage of time. In scheduler run 3, B and X had time remaining, and B ran because it had the highest priority.

During scheduler run 4, class X ran because class B had consumed its percentage allotment.

In a system where processes need more processing time than is available, the time interval tells the scheduler when to restart percentage counting. This may be important where a class of interactive user processes often exhausts its percentage (as mentioned earlier); in such cases you can shorten the interval.

Job processor time is allotted to a class, not to individual processes in that class. During scheduler runs 1 and 2, only the priority 1 process in class A got time; the priority 3 process got no time because the other process was higher priority and always ready to run. If you have processes that need specific job processor time allotment, they should either be in their own class or be in a class where other processes won't consume the entire time allotment.

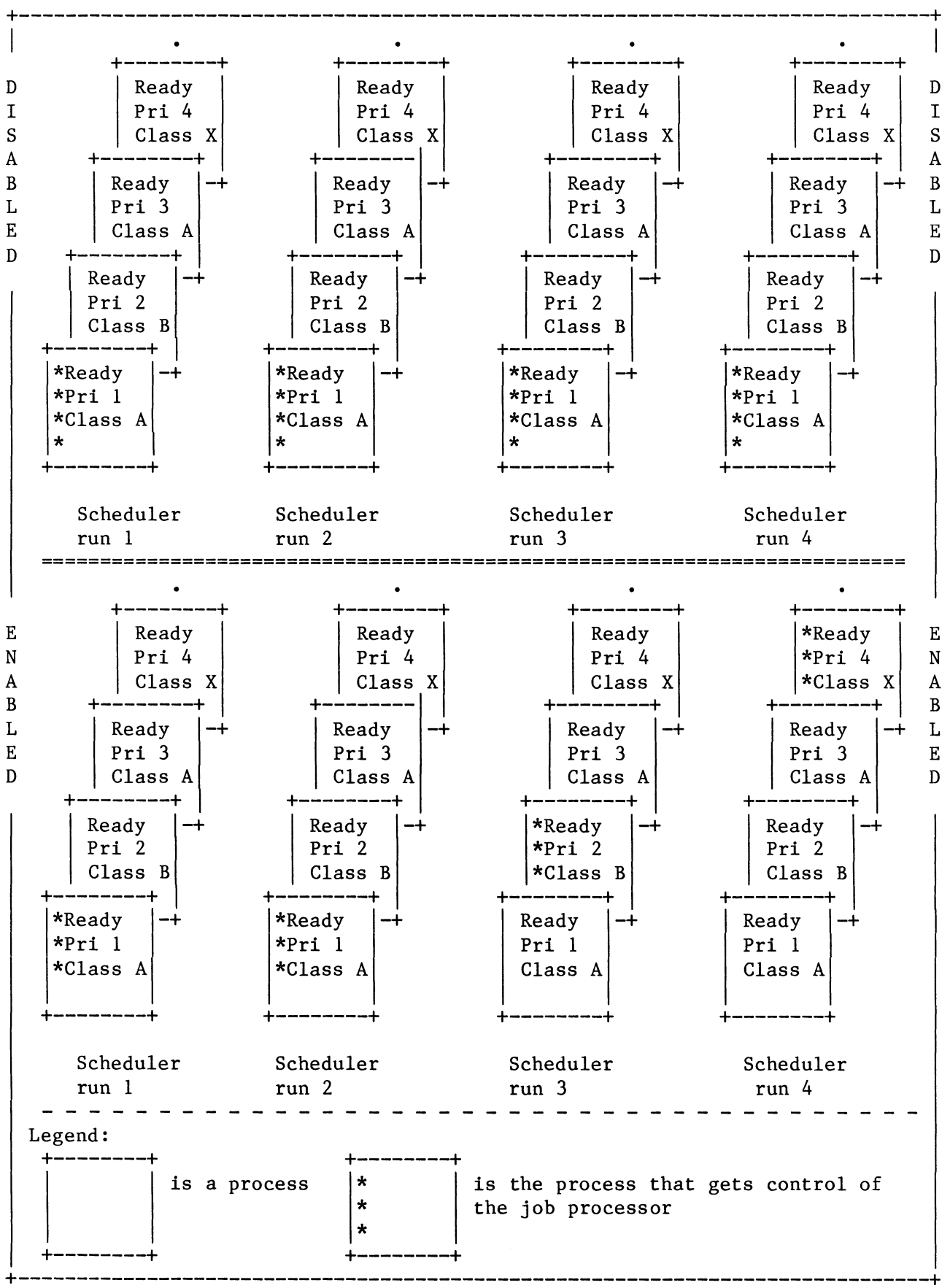

Figure 1-5. Processes in Standard and Class Scheduling

Classless Processes

If your computer has more than one job processor, be aware that certain processes will always run in the default job processor (called the mother processor) no matter where you try to run them. These classless processes include any process that does system tasks (like the PMGR) and any process that's issued call ?IDEF to define a user device. Processes that use ?IDEF include XTS or X25 and the global sync manager (GSMGR). Also, user programs that are privileged to access devices can issue ?IDEF. Specifying a job processor other than the mother for a classless process will do no harm -- but AOS/VS will ignore the definition and run the process on the default processor.

You can tell which processes are mother-processor only by running PED with the /MPROCESSOR switch. You might also include other PED switches that relate to classes. For example, you might start PED using the following switches.

```
PED/CYCLE=10/PID/USER/PROGRAM/ELAPSED/CPU/BS/ULOCALITY/PLOCALITY&<nl>
/CLASSNAME/IO/FTA/MPROCESSOR
```

(This omits only the /PROCESS, /SH7, /US7, and /WSS switches from the default PED display.)

The resulting PED display might look like Figure 1-6, next.

PID	USERNAME	PROGRAM	ELAPS	CPU	MP	UL	PL	CLASSNAME	I/O	FTA
1	PMGR	PMGR	2-06	78:02	Y	0	0	DEFAULT.CL	58	9389
3	OP	EXEC	2-06	22.55		0	0	DEFAULT.CL	8686	1327
4	OP	XLPT	2-06	20.40		0	0	DEFAULT.CL	761	46
9	OP	PED	5.00	0.20		0	0	DEFAULT.CL	0	57
12	SALLY	CEO_CP	4/17	13.34		1	0	PRIVILEGED	690	6441
13	MARC	CEO_CP	1/23	3.02		0	0	DEFAULT.CL	155	1441
14	JACK	F77	20.00	0.37		0	0	DEFAULT.CL	9	122
45	SBK	CEO_CP	2/45	7.81		1	0	PRIVILEGED	504	5667
.	.	.	.	.		.	.	.	.	.

Figure 1-6. Sample PED Display with Classes

This display shows that the PMGR is a mother-only process. (Also notice that PED truncates class names to 10 characters.)

About CLASP

CLASP is a menu-driven program. You run it by selecting menu choices and typing commands.

CLASP works by making system calls to AOS/VS. These settings change the scheduling environment. The settings last only while AOS/VS runs: they vanish at shutdown, and must be made again -- if desired -- at the next AOS/VS startup.

Therefore, after deciding what you want to do with CLASP, you'll probably want to create a CLASP script file. A script file allows you to automate the CLASP settings, via the command

```
X  CLASP/BATCH/SCRIPT_FILE=pathname
```

After deciding on CLASP settings that you like, you can insert appropriate XEQ CLASP commands in CLI macros -- the UP macro and/or macros to run at different times of day (as implied in Figures 1-2 through 1-4, earlier).

But before creating a useful script file, you'll probably want to create a class and logical processor environment, and try it out. For class scheduling to occur, you must do at least the following things with CLASP:

- create and/or modify one or more classes

- allot processor time to one or more classes

- enable class scheduling on the default logical processor

- apply settings or write a script file.

Other things you may want to do in CLASP are

- create one or more logical processors

- move a job processor to a different logical processor

- monitor class scheduling

- create an overview file of the CLASP environment.

You can do any of these things via an entry on the CLASP Main Menu. The Main Menu looks like Figure 1-7, following.

```
+-------------------------------------------------------------+
|              Class Assignment and Scheduling Package        |
|                                                             |
|     1   Create, delete or modify classes                    |
|     2   Create, delete or rename logical processors         |
|     3   Allot processor time to classes                     |
|     4   Change processor status                             |
|     5   Display processor hardware information              |
|     6   Apply CLASP settings to the operating system        |
|     7   Monitor logical processors                          |
|                                                             |
|     Enter choice                                            |
|                                                             |
|     For help, type ? or H              To exit, type Bye    |
+-------------------------------------------------------------+
```

Figure 1-7. CLASP Main Menu

There are several ways to use CLASP.

- explore and test (interactively, create and apply a class and logical processor environment)

- write a script file, after testing and exploring (from CLASP, using the WRITE command)

- run CLASP with script file (from the CLI, using the command X CLASP/BATCH/SCRIPT=script-file).

Table 1-1 lists menu choices you might make for different tasks.

Table 1-1. CLASP Menu Choices For Different Tasks

Test and Explore	Write Script File	Run Script File
1. Create...classes 2. Create...logical processors 3. Allot ... time to classes 4. Change processor status 5. Display...hardware info... 6. Apply CLASP settings... 7. Monitor logical processors	1. Create...classes 2. Create.. logical processors 3. Allot ... time to classes 4. Change processor status	No menus or interaction; CLASP applies all script file settings.

Class and Logical Processor Names

AOS/VS knows each class and logical processor only by its ID: a number between 0 and 15. CLASP maintains real names for classes and logical processors. You specify the name when you create the class or logical processor. Names are 1 to 16 characters long; all filename characters are allowed; and CLASP converts all lowercase letters to uppercase.

To maintain name-to-ID relationships, CLASP uses files in the peripherals directory (: PER). It creates a file in PER for each class and logical processor you create. This means that, to create classes and logical processors via CLASP, you need Superuser privilege.

CLASP Files and Requirements

The CLASP package includes the following files:

CLASP.CLI	(macro to execute program)
CLASP.PR	(program file)
CLASP.ST	(symbol table file, useful for patching if needed)
CLASP.RELEASE.NOTICE	(CLASP Release Notice, with features and warnings not described in this manual)

CLASP doesn't need information from any file shipped with AOS/VS, so you can install it wherever you want, without concern about search lists. CLASP creates all script and overview files in the working directory, unless you specify a different one.

Other programs that relate to class use include PREDITOR, SPRED, and PED, all shipped in :UTIL. This suggests that you may want to put CLASP in :UTIL. Or, you can put it in its own directory.

Normally, when you run CLASP, it assumes you want to change the current AOS/VS system. To do this, you need a special privilege (System Manager privilege) in your user profile. System Manager privilege entitles a user to create and delete classes and logical processors, initialize and release job processors, set the date and time, start and stop system logging, and other things. CLASP turns on System Manager mode exclusively -- which means that only one CLASP process with the power to change AOS/VS can run at a time.

Also, as mentioned above, CLASP's naming mechanism requires you to have Superuser privilege to create and delete classes and logical processors.

For unprivileged users, CLASP has a view-only mode (CLASP switch /VIEW_ONLY). In view-only mode, CLASP allows no changes, thus requires no privileges. View-only mode is designed to allow multiple processes to use CLASP's monitor and other informative features, while ensuring that only one CLASP process can change the system. (This is an important restriction, since two or more privileged CLASPs could overwrite one another's settings -- with potential for chaos in the multiuser environment.)

If you run CLASP with the /VIEW_ONLY switch and try to change anything, you'll receive the error message "This action is not allowed; you specified view only".

CLASP Command Line and Switches

Use the following command form to run CLASP.

```
                +---                                        ---+
                |   /SCRIPT_FILE=pathname  [ /BATCH ]       |
XEQ   CLASP     |                                           |
                |   /VIEW_ONLY                              |
                +---                                        ---+
```

Switch descriptions follow.

/BATCH

Tells CLASP to run noninteractively, without asking questions. You must also specify a script file (/SCRIPT_FILE switch). The /BATCH and /SCRIPT_FILE switches let you start a specific class and logical processor environment without interactive dialog -- ultimately useful via the UP macro. If you use both switches, CLASP applies the script file settings immediately, then terminates. A sample command line is

) X CLASP/SCRIPT=STD.CLASSES/BATCH <nl>

(<nl> stands for pressing the NEW LINE key)

/SCRIPT_FILE=pathname

Tells CLASP to use the settings stored in script file pathname. You can write a script file from CLASP via the CLASP command WRITE. If you use this switch with the /BATCH switch, CLASP will run without asking questions. If you omit /BATCH, CLASP will run interactively, but take its settings from the script file.

Using /SCRIPT_FILE without /BATCH is useful when you want to test a script file interactively or edit a script file or as a starting point for a different environment (add or change settings in the "old" script file).

/VIEW_ONLY

Runs CLASP in view-only mode. This requires no special privileges but allows no changes. It allows a user to look at any screen but not change anything. You must use this switch to run CLASP if

- a privileged CLASP process (one started without this switch) is running; or
- you lack System Manager privilege.

No other switch is allowed with /VIEW_ONLY.

Benefits of Class Scheduling

Class scheduling can be useful when specific processes need more job processor time. If, with your system under maximum load, job processor time remains available, class scheduling probably won't help improve performance.

You can use CLASP to check the amount of job processor time available. When your system is under maximum processing load, run CLASP and use its monitor screen to check the Idle figure. Execute CLASP, select the monitor choice, select logical processor DEFAULT.LP, then watch the Idle figure for a few minutes.

If the Idle figure shows a relatively large percentage of idle time (say 3% or more), CLASP probably can't help. If the Idle figure is less than 3%, CLASP may help.

Performance issues aside, you can use CLASP to enforce site rules and disciplines. For example, you can use it to restrict use of certain programs (like games or checkbook-balancing programs) during working hours. To prevent use of a program, set it up (with SPRED) to run in a nondefault class, then run a logical processor that allots the class no time. You can lift the ban by running a logical processor that does allot the class time.

Or, you can use CLASP simply as a monitor, without planning performance changes. For example, assume you have six departments and want to see what relative percentage of processor time each one is using. Just create a class for each department, run the CLASP monitor in its accumulation mode (not enabling class scheduling) and check the percentage figures.

Error Conditions

Syntax errors you make in CLASP will produce an error message near the bottom of the screen; then you can correct the bad entry.

CLASP has a Help facility that you can use by typing ? <nl> or HELP <nl>. It displays a list of CLASP commands that pertain to the current screen. Type HELP <nl> when you're unsure of the next step or after making an error.

CLASP applies its environment settings to AOS/VS using system calls. Interactively, CLASP settings aren't applied until you select choice 6, "Apply", from the Main Menu. Noninteractively, settings are applied when you run CLASP with the /SCRIPT_FILE and /BATCH switches. Therefore, any environment you create isn't implemented until you choose "Apply" or write and test a script file. Settings that might cause problems will have no effect until you test them. You should test CLASP settings in the pertinent environment -- if possible -- before creating a script file and relying on the settings as a matter of course.

You can leave CLASP by typing BYE <nl> from the Main Menu or any

secondary screen. However, unless you've applied changes ("Apply" choice) or created a script file, all settings within CLASP vanish when the program terminates. To save your changes (if you want), type WRITE <nl> and create a script file, and/or use the "Apply" choice.

It's possible with CLASP to define an environment that deprives processes of processor time. This is called stranding processes. Stranding processes and other class-related errors are explained in Chapter 4.

End of Chapter

Chapter 2

Using CLASP

This chapter describes using CLASP: features, Main Menu, and other screens and functions. Read it whenever you want specific information about CLASP.

Many of the major sections in this chapter are based on the CLASP Main Menu. The major sections are

- Using Classes -- When and How
- Running CLASP

- The Main Menu
- Create, Delete, or Modify Classes -- Choice 1
- Create, Delete, or Rename Logical Processors -- Choice 2
- Allot Processor Time to Classes -- Choice 3
- Change Processor Status -- Choice 4
- Display Processor Hardware Information -- Choice 5
- Apply CLASP Settings to the Operating System -- Choice 6
- Monitor Logical Processors -- Choice 7

- Creating and Using Script and Overview Files
- Security Issues

Using Classes -- When and How

This section outlines the steps that lead up to -- and include -- using classes. It will help you understand when and how to use class scheduling.

1. Define your applications environment without classes. This includes installing hardware and AOS/VS, and specifying, coding, and debugging your initial application programs.

2. Run the applications environment for awhile.

3. Encounter some limit within standard scheduling. Identify it and decide that classes and logical processors can help you. If there's no limit that class scheduling can help overcome, don't bother with class scheduling.

4. Set up the class environment. This means planning classes and logical processors, and implementing them as follows:

   a. If a class will be based on username, use PREDITOR.

   b. If a class will be based on program name, use SPRED.

      A class can be based on a combination of username and program name.

c. Create classes and logical processors with CLASP. Allot percentages of processing time to classes. When you allot percentages, use a tentative figure like 100%. CLASP's monitor will give you better figures later. While you're in CLASP, tell it to accumulate class statistics. Then create a script file, to recreate these acts easily later.

5. Start CLASP with the script file created above; then run your typical applications environment.

6. Use CLASP monitor information to refine class percentages, other class parameters, and/or class-to-logical processor assignments via CLASP, PREDITOR, or SPRED.

   To give a class more time than it's currently getting, use CLASP to give it a larger percentage than CLASP's monitor reports. To give a class less time, give it a smaller percentage.

7. Using the CLASP monitor, compare the percentages used by classes to your goals for those classes.

   If the reported percentages are not close to your goals, return to step 6. If the percentages are close to your goals, continue.

8. Use CLASP interactively to enable class scheduling. (Thus far, you've simply been modeling a class environment.)

9. See if the environment (interactive and noninteractive processes) is satisfactory.

   If it's not satisfactory, use CLASP to change from enabled to accumulation mode. Return to step 6.

   If the class environment is satisfactory, create a CLASP script file and arrange to have CLASP run the script automatically, perhaps via an UP macro. You're done!

These steps may seem formidable, but they represent the most productive way to approach the class environment. This chapter deals primarily with CLASP operations information (steps 4, 6, and 8). But the other steps are important conceptually.

Running CLASP

To make changes with CLASP, you must have the System Manager privilege in your user profile. (This privilege allows a user to initialize job processors with the JPINITIALIZE command, and to create classes and logical processors.) You must also have Superuser privilege, since CLASP creates files in the peripherals directory (:PER).

When you want to make changes with CLASP, omit the /VIEW_ONLY switch. Without /VIEW_ONLY, CLASP assumes that you want to make changes. CLASP tries to turn on System Manager privilege exclusively, which prevents any other CLASP process without /VIEW_ONLY from running. This prevents

more than one CLASP process at a time from changing the system.

Multiple users -- who don't need System Manager or Superuser privilege -- can view CLASP settings and the CLASP monitor by using the /VIEW_ONLY switch. When started with /VIEW_ONLY, CLASP doesn't try to use any privilege. It allows users to look at any screen, but not change anything. They need only execute access to the CLASP program file. If you run CLASP with the /VIEW_ONLY switch and try to change anything, you'll receive the error message "This action is not allowed; you specified view only".

The classes and logical processors defined by CLASP remain in force only while AOS/VS is running. When AOS/VS shuts down, all classes and logical processors disappear. Thus, you'll probably want to use a CLASP script file -- perhaps in a macro called from the UP macro -- to create classes and logical processes automatically when you bring up AOS/VS.

The format of the CLASP command line is

```
XEQ  CLASP    [   /SCRIPT_FILE=pathname  [/BATCH]    ]
              [   /VIEW_ONLY                         ]
```

Switch /SCRIPT_FILE=pathname tells CLASP to take environment settings from the script file; use it when you don't want to supply settings from scratch. Switch /BATCH -- used only with /SCRIPT_FILE -- tells CLASP to apply the script settings without asking questions. Both switches are needed for noninteractive operation.

Switch /VIEW_ONLY runs CLASP in view-only mode. This is needed for users who don't have special privileges or for any user if a CLASP that can change AOS/VS is already running.

All three switches are detailed in Chapter 1.

The Main Menu

CLASP starts by displaying its Main Menu, which looks like Figure 2-1.

```
+-------------------------------------------------------------------------+
|                 Class Assignment and Scheduling Package                 |
|                                                                         |
|                              Main Menu                                  |
|                                                                         |
|       1   Create, delete, or modify classes                             |
|       2   Create, delete, or rename logical processors                  |
|       3   Allot processor time to classes                               |
|       4   Change processor status                                       |
|       5   Display processor hardware information                        |
|       6   Apply CLASP settings to the operating system                  |
|       7   Monitor logical processors                                    |
|                                                                         |
| Enter choice                                                            |
|                                                                         |
| For help, type ? or Help                          To exit, type Bye     |
+-------------------------------------------------------------------------+
```

Figure 2-1. The CLASP Main Menu

Choice 1 (called "Create ... classes" for brevity) lets you create, delete, or modify a class; it's a required step for class scheduling. Choice 2 (called "Create ... processors") lets you create, delete, or rename logical processors.

Choice 3 ("Allot...") lets you allot logical processor time to classes. Until you do this, only the default class (with all processes) gets job processor time. This is a required step for class scheduling.

Choice 4 ("Change processor status") tells AOS/VS to enable or disable class scheduling, or to accumulate statistics to monitor processors. Choice 4 also allows you to move a job processor to a different logical processor. Accumulating statistics provides useful feedback on your class definitions; enabling class scheduling is required for class scheduling; moving job processors around is very useful.

Choice 5 ("Display processor ... information"), displays job processor hardware information, like whether the floating-point is enabled.

Choice 6 ("Apply") tells CLASP to apply all its settings to the operating system. Applying settings is a required step. When you're running CLASP interactively, you must use this choice to apply. When you run CLASP with a script file and /BATCH switch, it applies the script file settings automatically.

Choice 7 ("Monitor") lets you monitor logical processors. CLASP displays the percentage of time used by each class. Monitoring gives you a starting place for, and feedback on, class definitions. Monitoring is most useful when CLASP is accumulating statistics (choice 4, "Change processor status") and settings have been applied (choice 6, "Apply").

Commands to CLASP

CLASP accepts the following commands. You can abbreviate each command; for example, you can type B instead of BYE.

Command	Meaning
?	Help. Lists commands and options.
BYE	Exits from CLASP to the CLI. If you've changed anything during this CLASP run and want to create a script or overview file, use the WRITE command before using BYE. To apply all CLASP settings -- including changes -- to the system, select choice 6 "Apply", then exit with BYE if desired. If you don't create a script or overview file, or apply settings, changes you made will be lost when CLASP terminates.
HELP	Help. Lists commands and options.
REFRESH	Refreshes the screen display -- useful after someone has sent you a message.
WRITE	Tells CLASP to write a script or overview file. The program asks whether you want a script or overview file, then it asks for the pathname. A script file is a description of all CLASP settings -- the the settings you've changed during this CLASP session (if any) and the settings you haven't changed. A script file allows CLASP to run without interaction, perhaps via a CLI macro that can be started routinely by your system UP macro. An overview file summarizes current CLASP environment settings. You can print this and keep it handy near the computer, for easy reference. Script and overview files are detailed near the end of this chapter, in "Using Script and Overview Files".
BREAK/ESC key	Escapes from the current screen or question. At any prompt, this key tells CLASP to cancel the current query and return to the previous one. It allows you to back out of a dialog and/or return to the Main Menu from a lower level screen. BREAK/ESC works the same way as the CANCEL/EXIT key in DG's CEO (Comprehensive Electronic Office) system. (If you're using the system console, the keys CMD and BREAK/ESC give control to the SCP CLI. But BREAK/ESC by itself -- without CMD -- serves as an escape within CLASP and does not give control to the SCP.)

Create, Delete, or Modify Classes -- Choice 1

Choice 1 allows you to create, delete, or modify a class. After you select choice 1, CLASP displays the appropriate screen. The default screen, before any classes have been created, looks like Figure 2-2.

```
                Create, delete, or modify classes

                                Program Locality
                                                                  Class Name Key
          0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
        ----------------------------------------------------
     0| A  A  A  A  A  A  A  A  A  A  A  A  A  A  A  A |  A = DEFAULT.CLASS
     1| A  A  A  A  A  A  A  A  A  A  A  A  A  A  A  A |  B =
U    2| A  A  A  A  A  A  A  A  A  A  A  A  A  A  A  A |  C =
s    3| A  A  A  A  A  A  A  A  A  A  A  A  A  A  A  A |  D =
e    4| A  A  A  A  A  A  A  A  A  A  A  A  A  A  A  A |  E =
r    5| A  A  A  A  A  A  A  A  A  A  A  A  A  A  A  A |  F =
     6| A  A  A  A  A  A  A  A  A  A  A  A  A  A  A  A |  G =
L    7| A  A  A  A  A  A  A  A  A  A  A  A  A  A  A  A |  H =
o    8| A  A  A  A  A  A  A  A  A  A  A  A  A  A  A  A |  I =
c    9| A  A  A  A  A  A  A  A  A  A  A  A  A  A  A  A |  J =
a   10| A  A  A  A  A  A  A  A  A  A  A  A  A  A  A  A |  K =
l   11| A  A  A  A  A  A  A  A  A  A  A  A  A  A  A  A |  L =
i   12| A  A  A  A  A  A  A  A  A  A  A  A  A  A  A  A |  M =
t   13| A  A  A  A  A  A  A  A  A  A  A  A  A  A  A  A |  N =
y   14| A  A  A  A  A  A  A  A  A  A  A  A  A  A  A  A |  O =
    15| A  A  A  A  A  A  A  A  A  A  A  A  A  A  A  A |  P =

  Enter choice (Create, Delete, Modify, Substitute)
```

Figure 2-2. Create, Delete or Modify Classes Screen

The table bordered by "User Locality" and "Program Locality" is called the class matrix. This matrix has 16 user and program localities. At the right are class name keys and class names. Within the matrix, you define the scope of each class by specifying one or more locality pairs -- a user locality and program locality. All processes running in the intersecting user and program localities will belong to that class.

The figure shows the AOS/VS default class. No new classes have been created, so there is only one class. The class name key is A, and the class name is DEFAULT.CLASS. The default class includes all locality pairs -- 16 user and 16 program -- which means all processes will run in this class.

The letters in the matrix (initially all A, can be A through P) stand for class IDs. AOS/VS uses numbers 0 through 15 for class IDs; but CLASP uses letters to help distinguish the classes from the localities (otherwise, both would appear as numbers -- confusing).

Commands you can use with this screen include those shown earlier and others, like CREATE, MODIFY, and DELETE.

Here's a list of all commands you can use with the class creation screen. You can abbreviate any command.

Command	Meaning
?	Help. Lists commands.
BYE	With all screens, the BYE command exits to the CLI. Unless you've created a script or overview file (WRITE command) or applied changes (choice 6 "Apply"), any changes you made will be lost when CLASP terminates.
CREATE	Creates a class. The new class can't already exist. Existing classes have names following the = sign in the right column. (If you want to modify a class, use MODIFY.) After you type CREATE, CLASP asks Specify new class name: Class names can be up to 16 valid filename characters. Make each name as descriptive as possible, for easy reference in the future. (The first 10 characters are particularly important, since PED can display only 10 characters of a class name, and you will probably use PED in the future to discover the class of a process.) CLASP converts all class names to uppercase (you can specify them in any case). If you make a mistake typing a class name, use the DEL key to erase, then type the correct name. When you're satisfied with the name, press <nl>. For example, if you plan a class for privileged people, you might type PRIVILEGED.USER <nl> After you type the name and <nl>, CLASP displays the name on the screen, next to the first available name key. You'll use this name key to reassign locality pairs to the new class. Then, CLASP asks Do you want to create another class (Y or N)? [Y] The idea here is to create all your classes first, then reassign locality pairs for all using MODIFY. If you want to do classes one by one (create a class, reassign locality pairs to it, create another class, reassign to it, etc.), type N <nl>. CLASP will ask about locality pairs. If you want to create all classes, then reassign all locality pairs, press <nl> for the default. CLASP will then ask again for a class name.

Command	Meaning

CREATE (continued)

After creating all the classes you want, type N <nl> to this "...create another class..." question. CLASP then asks

Do you want to reassign locality pairs (Y or N)? [Y]

A class has no meaning without locality pairs, so you will probably want to reassign locality pairs to all the classes you created. (We use the term "reassign" because all locality pairs are always assigned to some class -- all pairs start life assigned to the DEFAULT.CLASS.)

To reassign locality pairs, press <nl>. If you don't want to do this now, type N <nl>; you can reassign later using the MODIFY command.

If you say No, the cursor returns to the "Enter choice" prompt. You can return to the Main Menu via BREAK/ESC, then proceed to another choice (like 6 "Apply" or 4 "Change processor status"); or you can leave CLASP.

If you press <nl>, CLASP asks about localities as follows:

```
Specify class name key:
Specify user locality:        Specify program locality:
Do you want to reassign another locality pair (Y or N)? [Y]
```

Answering these questions is explained below, under "How to Reassign Locality Pairs".

DELETE

Deletes a class. The class to delete must exist (shown by a name following the equals sign in the right column). If you want to modify a class, use the MODIFY command, not DELETE.

You can't delete (or modify the name of) the default class.

You can't delete any class unless all its locality pairs are reassigned to other classes. (The class name key must be absent from the class matrix.) This rule prevents processes that belong to a class from becoming undefined when the class vanishes.

When you try to delete a class that has locality pairs, CLASP asks for the name key of a class to substitute for it. If you want to continue with the deletion, type a valid name key. CLASP will then add the target class's locality pairs to the substitute's pairs and delete the target class. If you don't want to delete, press <nl>; CLASP will then cancel the delete sequence and return the "Enter choice" prompt.

To substitute, you may want to use a new class that has no locality pairs defined. This lets the substitute class keep a copy of the target class's locality pairs (instead of having the substitute's pairs combined with the target's pairs).

Command	Meaning
DELETE (continued)	After you type DELETE, CLASP asks Specify class name key: Type the name key of the class you want to delete, followed by <nl>; for example, E <nl>. Then, if this is a valid class and its localities are reassigned to other classes, CLASP deletes the class. Its name vanishes from the screen.
HELP	Helps by listing commands.
MODIFY	Modifies a class. The class must already exist, as shown by a name following the appropriate = sign in the right column. This command allows you to change a class's locality pairs or name. (But you can't modify the name of the default class.) After you type the MODIFY command, CLASP asks Modify locality pairs or name (Locality or Name)? [Locality] To change locality pairs, press <nl>; to change a class name, type N (or NAME) and <nl>. Then, CLASP asks Specify class name key: Type the letter name key of the class you want to modify, followed by <nl>; for example, B <nl>. Next, depending on the action you chose above, CLASP asks either Specify user locality: Specify program locality: Do you want to reassign another locality pair (Y or N)? [Y] or Specify new class name [xxxx] (xxxx is the old name) For locality pairs, specify the user locality, then program locality -- explained below under "How to Reassign Locality Pairs". For the name, type the desired new name after the old one and press <nl>. (You can retain the old name by pressing <nl>). CLASP changes the name display to reflect any change you make.
REFRESH	Refreshes the screen.
SUBSTITUTE	Substitutes one class's locality pairs for another class's pairs. This command is useful when you want to delete a class, yet keep its locality pairs (in another class) for future use or reference. When you substitute, CLASP moves the source class's locality pairs to the substitute class -- adding them to any pairs the substitute already has.

Command	Meaning
SUBSTITUTE (continued)	CLASP asks about the source and destination classes as follows: Specify name key of source class (whose pairs you want moved): and Specify name key of destination class (to receive the pairs): After you type valid name keys, CLASP gives the source class's locality pairs to the destination class -- removing all occurrences of the class name key from the source class.
WRITE	Tells CLASP to write a script or overview file. The program asks whether you want a script or overview file; then it asks for the pathname and ultimately creates the file. Whenever you create a script or overview file, it contains the entire environment within CLASP. This includes all items you specified during this CLASP run and the defaults for items you didn't specify. Creating a script file is useful whenever you want to save current CLASP settings. For a script file to create a class environment, classes must be allotted processor time (choice 3 "Allot") and class scheduling must be enabled (choice 4 "Change processor status"). If you want to use CLASP's monitor, use choice 4 "Change processor status" to turn on accumulation mode instead of enabling scheduling.
BREAK/ESC key	Escapes from the current screen or question. At any prompt, this key tells CLASP to cancel the current query and return to the previous one. At "Enter choice", it returns to the Main Menu, where you can select another choice (like 2 "Create ... processors", 3 "Allot", or 4 "Change processor status"), or exit with BYE.

For every class you create, CLASP maintains a file in the peripherals directory (:PER). This file associates the class name you specified to CLASP with the class ID (a number AOS/VS uses to identify the class). The file isn't created until you apply settings (choice 6 "Apply").

About User and Program Locality

The user and program locality pairs you specify to CLASP identify the class of any process that matches them.

A class is defined by one or more locality pairs: an intersection of user and program localities. For example, say you create a class called PRIVILEGED.USER and specify user localities 2 and 3 -- and all program localities -- for it. This produces pairs (intersections) of user localities 2 and 3 with all program localities. Thus, any process in user localities 2 or 3 (regardless of program locality) will belong

to the PRIVILEGED.USER class.

Each user process is created with the default user locality specified in the PREDITOR profile. The PREDITOR default locality is 0. If the PREDITOR default user locality for a user is 2 or 3, then the user process will be created in the PRIVILEGED.USER class. By default, all its sons will also belong to the PRIVILEGED.USER class. If the PREDITOR default user locality is anything but 2 or 3, the user process will not be created in the PRIVILEGED.USER class.

If a user's PREDITOR profile includes the "Use other localities" privilege, he or she can change user locality with the CLI command LOCALITY -- or create a process of a different locality with the PROCESS command /LOCALITY switch. Changing user locality may or may not change the process class. In the example above, changing user locality from 0 to 1 wouldn't change the process' class -- but changing user locality from 0 to 2 would change its class.

Program locality can't be changed by CLI commands; it can be changed only via the SPRED program. Edit the program file with SPRED; then run the program. Since program locality is checked before the process is created, changing a program's locality doesn't affect currently running processes from that program file.

How to Reassign Locality Pairs

The user and program localities you specify to CLASP define each class (although you must also use choice 3 "Allot" from the Main Menu to allot processor time to the class).

You can reassign locality pairs when you create or modify a class. When you do reassign locality it, CLASP asks

```
Specify class name key:
Specify user locality:          Specify program locality:
Do you want to reassign another locality pair (Y or N)? [Y]
```

The specify questions define the class in which a process with the given locality pairs will run. CLASP repeats all these questions -- allowing you to specify multiple pairs as desired -- until you tell it that you're done by typing N <nl>.

When you reassign pairs, CLASP displays the name key bright, while unchanged name keys remain dim. This makes it easy to see which user and program locality(ies) you just reassigned.

Valid answers are integers 0 through 15, or <nl> to specify 0 through 15. When CLASP asks about user locality, an answer of <nl> specifies all users (user localities 0 through 15); when CLASP asks about program locality, <nl> specifies all programs (program localities 0 through 15).

Thus, you can define the same class for one user locality -- including all programs -- by typing a number for the user locality, then <nl> for the program locality. This is useful, for example, for a class destined

for specific users, regardless of the programs they run.

And, you can define the same class for one program locality -- including all users -- by pressing <nl> for user locality and typing a number for program locality. This is useful when a class is destined for one or more specific programs (perhaps for programs like compilers that you want run as a batch jobs), regardless of user.

If you answer <nl> for both user and program locality, the class will include all processes -- as does the original default class. Generally, don't do this, because it can leave essential processes -- like your CLASP process -- without processing time on a logical processor.

If you accidentally do press <nl> at a ''...locality'' question (filling part or all of the matrix with one name key), don't panic. The matrix is a temporary entity until you apply settings. You can restore the original settings by leaving CLASP and executing it again, with the original command line.

Why Use This Screen?

This screen is essential to class scheduling: for class scheduling to work, you must specify at least one locality pair for one class (the default class). Nearly always, you'll want to create other classes and specify locality pairs for them.

Also, you may want to modify existing locality pairs or class names -- perhaps those in a script file, with a view toward creating an updated script file.

This screen allows you to perform both operations.

Generally, after creating classes and reassigning locality pairs, your next step with CLASP is to create a logical processor or allot processor time to the new class(es) -- via Main Menu choice 2 "Create ... processors" or 3 "Allot".

Class Creation Example

In this example, you want to create a class for privileged users. You want the new class to include any program run by users with a user locality of 2 or 3 -- regardless of program locality. You decide to name the class PRIVILEGED.USER, as mentioned above. The dialog might resemble that in Figure 2-3, next.

Console Dialog	Explanation of Dialog
) X CLASP <nl>	(Start CLASP)
.	(It displays Main Menu)
.	
Enter choice 1 <nl>	(Select "Create" choice)
Create, delete, or modify classes	(CLASP displays "Create"
.	screen; the default
.	class, with name key A,
.	fills the class matrix)
Enter choice (Create, Delete ... CREATE <nl>	(Choose "Create")
Specify new class name: PRIVILEGED.USER <nl>	(Specify class name)
Do you want to create another class...[Y] N <nl>	(No, not another)
Do you want to reassign locality pairs...[Y] <nl>	(Yes, want to reassign)
Specify class name key: B <nl>	(The privileged class)
Specify user locality: 2 <nl>	(The user locality is 2)
Specify program locality: <nl>	(Use <nl> to specify all program localities, 0 through 15)
Do you want to reassign another locality pair (Y or N)? [Y] <nl>	(Yes, reassign another)
Specify class name key [B]: <nl>	(Continue with class)
Specify user locality: 3 <nl>	(The user locality is 3)
Specify program locality: <nl>	(Again, use <nl> to specify all localities, 0 through 15)
Do you want to reassign another locality pair (Y or N)? [Y] N <nl>	(No, done with locality pairs)
Enter choice (Create, Delete, Modify ...)	(Original prompt returns and the class matrix appears as follows)

Program Locality

Class Name Key

User locality	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Class Name Key
	0	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A = DEFAULT.CLASS
	1	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	B = PRIVILEGED.USER
U	2	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	C =
s	3	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	D =
e	4	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	E =
r	5	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	F =
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.

Figure 2-3. Creating a Class

Modifying Existing Classes

After you've created a class and specified localities for it, you might want to change the class's locality pairs or name. You can change or add locality pairs for a class or change a class name via the MODIFY comand. The questions are the same as above, in "How to Reassign Locality Pairs". (To remove a locality pair from a class, you must reassign the pair to another class -- possibly to the default class).

For example, assume you created the class PRIVILEGED.USER shown above. You've been using classes for a while and you want to change this class as follows:

- change class name PRIVILEGED.USER to PRIVILEGED.USERS

- reassign a locality pair -- user locality 1 and program locality 1 -- to the class

The dialog might go as in Figure 2-4, next.

```
Console Dialog                                          Explanation of Dialog
--------------                                          ---------------------

) X  CLASP <nl>                                         (Start CLASP)
.                                                       (It displays Main Menu)
.
Enter choice  1 <nl>                                    (Select "Create" choice)
Create, delete, or modify classes                       (CLASP displays "Create"
.                                                       screen for current
.                                                       environment)
.

Enter choice (Create, Delete ...  MODIFY <nl>           (Want to modify)
Modify locality pairs or name
        (Locality or Name)? [Locality]  NAME <nl>       (New name for class)
Specify class name key:  B <nl>                         (Identify the class)
Specify new class name [...  PRIVILEGED.USERS <nl>      (Specify new class name)
Enter choice (Create, Delete ...  MODIFY <nl>           (Still need to modify
                                                        locality pairs)

Modify locality pairs or name
        (Locality or Name) ? [Locality]  <nl>           (Choose locality pairs)
Specify class name key:  B <nl>                         (Identify the class)
Specify user locality:  1 <nl>                          (Add user locality 1)
                Specify program locality:  1 <nl>       (Add program locality 1)

Do you want to reassign another locality
                 pair (Y or N)? [Y]  N <nl>             (Say no; the class
                                                        matrix appears as
                                                        follows)
```

```
                        Program Locality
                                                              Class Name Key
        0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
      ---------------------------------------------------
   0 | A  A  A  A  A  A  A  A  A  A  A  A  A  A  A  A | A = DEFAULT.CLASS
   1 | A  B  A  A  A  A  A  A  A  A  A  A  A  A  A  A | B = PRIVILEGED.USERS
U  2 | B  B  B  B  B  B  B  B  B  B  B  B  B  B  B  B | C =
s  3 | B  B  B  B  B  B  B  B  B  B  B  B  B  B  B  B | D =
e  4 | .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  . | E =
r  5 | .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  . | F =
   . | .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  . | .
```

Figure 2-4. Modifying a Class

Saving Your Efforts

Unless you started CLASP with the /VIEW_ONLY switch or are experimenting, you'll probably want to preserve the work you do in CLASP.

CLASP settings can be applied to the system two ways:

- Interactively, when you select choice 6 "Apply" on the Main Menu; or

- Noninteractively, when you start CLASP with a script file and the /BATCH switch.

CLASP settings are not applied to the system until one of these events occurs.

CLASP environment settings aren't saved on disk until you type WRITE <nl> and specify either a script or an overview file. This means -- unless you were just experimenting with CLASP -- you'll want to apply settings or create a script or overview file sometime during a CLASP session.

Other Steps with CLASP

Creating classes and reassigning locality pairs is only one step toward establishing your own useful class environment.

Before you proceed to apply settings or create a script or overview file, think about other steps with CLASP. You can

- create one or more logical processors;

- allot processor time to classes (a required step for class scheduling to work);

- change processor status (turn on accumulation mode to monitor, move job processors around, or enable class scheduling -- the last a required step);

- apply settings (a required step to activate class scheduling interactively).

There's a detailed example of class use in Chapter 3.

Create, Delete, or Rename a Logical Processor -- Choice 2

Choice 2 lets you create, delete, or rename logical processors. At AOS/VS startup, there is just one logical processor, DEFAULT.LP. The default job processor is connected to this logical processor.

A logical processor is a scheduling arrangement that consists of classes. One or more job processors can be connected to it. All logical processors except the default exist only while AOS/VS runs. All except the default are deleted when AOS/VS shuts down -- they must be created again, if desired, when AOS/VS is brought up again. (Even DEFAULT.LP reverts to the original DEFAULT.CLASS setting at shutdown.)

Up to 16 logical processors can exist at one time. But class scheduling occurs only on processors that

- have time allotted to classes that have locality pairs defined,

- have one or more job processors connected, and

- have class scheduling enabled.

You can name (or rename) a logical processor (except DEFAULT.LP) by the same rules as a class. Up to 16 filename characters are allowed.

When you type 2 <nl> from the Main Menu, CLASP displays a list of logical processors, then asks what you want to do. The original list shows only the default processor, DEFAULT.LP. The list looks like this:

```
+-----------------------------------------------------------------+
|                                                                 |
|            Create, delete, or rename logical processors         |
|                                                                 |
|   1  DEFAULT.LP                                                 |
|                                                                 |
~                                                                 ~
|   Enter choice (Create, Delete, or Rename)                      |
|                                                                 |
+-----------------------------------------------------------------+
```

Specify what you want to do -- create, delete, or rename.

Commands you can use with this screen include global CLASP commands and CREATE, DELETE, and RENAME. On CREATE, CLASP asks for the name. On DELETE and RENAME, it asks for the menu number of the processor entry (this is 1 for the default processor). You can't rename or delete the default processor, DEFAULT.LP.

Here's a list of all commands you can use with this screen. You can abbreviate any command.

Command	Meaning
?	Help. Lists commands.
BYE	With all screens, the BYE command exits to the CLI. Unless you've created a script or overview file (WRITE command) or applied changes (choice 6 "Apply"), any changes you made will be lost when CLASP terminates.
CREATE	Creates a logical processor. The new processor can't already exist. After you type the CREATE command, CLASP asks you to Specify the logical processor name: A logical processor name can have up to 16 filename characters. You should make each name as descriptive as possible, for easy reference in the future. Use a name that summarizes what you want the logical processor to do or when you plan to use it. CLASP converts all characters to uppercase, although you can type them in any case. For example, if you plan a logical processor to run at night, you might type NIGHT.LP <nl> After you type the name and <nl>, CLASP displays the name on the screen and returns to the "Enter choice..." prompt. Each new logical processor is created with the same settings as the default processor's defaults (DEFAULT.LP's defaults). These are: class name DEFAULT.CLASS with name key A, all locality pairs of the default class (key A), and definition as a primary class allotted 100% of processor time.
DELETE	Deletes a logical processor. You can delete a logical processor system only if there are no job processors connected to it. You can't delete the default processor, DEFAULT.LP. You can use choice 4 "Change processor status" to move any connected job processor to another logical processor; then delete the logical processor. After you type DELETE, CLASP asks Specify LP menu number: Type the number that corresponds to the logical processor, followed by <nl>. Then, if no job processor is connected to it, CLASP deletes the logical processor. Its name vanishes from the screen.
HELP	Helps by listing commands.
REFRESH	Refreshes the screen.

Command	Meaning
RENAME	Renames a logical processor. CLASP asks for the processor menu number, as with the DELETE command. Then it asks for the new name: Specify the new processor name [xxxx] (xxxx is the old name) Type the new logical processor name -- 16 or fewer characters -- and press <nl>. Or, to retain the old name, press <nl>. CLASP updates the screen with the new processor name. You cannot rename the default processor, DEFAULT.LP.
WRITE	Tells CLASP to write a script or overview file. The program asks whether you want a script or overview file, then it asks for the pathname and ultimately creates the file.
BREAK/ESC key	Escapes from the current screen or question. At any prompt, this key tells CLASP to cancel the current query and return to the previous one. At "Enter choice", it returns to the Main Menu, where you can select another choice (like 3 "Allot", or 4 "Change processor status"), or exit with BYE.

For every logical processor you create with CLASP, CLASP maintains a file in the peripherals directory. This file associates the processor name you specified to CLASP with the processor ID (a number AOS/VS uses to identify the processor). The file isn't created until you apply settings (choice 6 "Apply").

Why Use This Screen?

Logical processors are very important parts of class scheduling. Each one is a different scheduling definition that you can activate quickly and easily via CLASP -- at any time.

Creating logical processors isn't strictly required, since you can always work with the default processor, DEFAULT.LP. However, to make good use of class scheduling, you'll probably want to create logical processors. And, if you create them, you may want to delete or rename them.

This screen allows you to do all of these things.

Generally, after creating one or more logical processors, the next step is to allot classes time on the new processor(s), via choice 3 "Allot" on the Main Menu. Then, you can move a job processor to the new logical processor, turn accumulate mode on (both operations are on 4 "Change processor status"), monitor the new processor, make additional changes, enable class scheduling, and create a script or overview file.

Example of Creating a Logical Processor

Figure 2-5 shows dialog that creates logical processor "FAVORS.BATCH".

Console Dialog	Explanation of Dialog
) X CLASP <nl>	(Start CLASP)
.	(It displays Main Menu)
.	
.	
Enter choice 2 <nl>	(Select "Create ... processors" choice)
	(CLASP displays "Create processors" screen)
Create, delete, or rename logical processors	
1 DEFAULT.LP	
Enter choice (Create, Delete... CREATE <nl>	(Want to create)
Specify the logical processor name: FAVORS.BATCH <nl>	(Type the name)
	(CLASP displays updated screen)
Create, delete, or rename logical processor	
1 DEFAULT.LP	
2 FAVORS.BATCH	
Enter choice (Create, Delete... BREAK/ESC	(Press BREAK/ESC to return to Main Menu)

Figure 2-5. Creating a Logical Processor

Allot Processor Time to Classes -- Choice 3

Choice 3 lets you allot processor time to classes, or change the current allotment. (A class isn't recognized until it's been allotted time on a logical processor.) Each class you specify must already exist. It can be the default class or a class you've created using choice 1, "Create ... classes". Choice 3 also allows you to change a processor's time interval (by default 4 seconds).

Allotting processor time includes several steps, as follows.

1. You must identify each class you want recognized by the logical processor. The logical processor will give time only to processes in these classes -- and to no others. To identify a class, you specify it as primary or secondary.

   Primary-class processes get preference over secondary-class processes. If two processes of equal priority compete, but one is primary and one secondary, the primary-class process gets the processor first. Secondary processes get time only if no primary-class processes are ready to execute.

   When you specify a class as primary, you will then indicate the percentage of time you want the class to get. This percentage is a maximum; the class will be allowed to use up to this amount. If a class doesn't use its full percentage, AOS/VS will give the time to other classes. A class can use more than its specified amount if no other class wants time.

   You can define more than one primary class. All primary classes are treated equally, until one of them consumes its percentage. The sum of percentages you specify for all classes should equal 100%, for reasons described below.

   Within primary classes whose percentage limit hasn't been reached, processes are scheduled in the standard way, by priority. This is true with logical processors in which class scheduling hasn't been enabled.

   For example, say you have four classes and allot them time on processor MY.PROCESSOR as follows:

   ```
   DEFAULT.CLASS      40%
   PRIVILEGED.USERS   30%
   BATCH.STD          20%
   BATCH.LOW          10%
   ```

   When class scheduling is enabled on MY.PROCESSOR, AOS/VS will try to give time proportionately as follows:

   ```
   DEFAULT.CLASS      4 units
   PRIVILEGED.USERS   3 units
   BATCH.STD          2 units
   BATCH.LOW          1 unit
   ```

When you specify a class as secondary, you will also indicate its level. There are 16 secondary levels, starting with level 1. Secondary classes on the same level compete as equals. Secondary classes on lower levels yield to higher level secondary classes. You can define all secondary classes on level 1. Or, you can define other levels.

Secondary classes get time only when no primary-class processes are ready to run -- regardless of primary class allotments. The system will always let a primary-class process run (even if the class has consumed its allotment) before letting a secondary process run.

For an example with both primary and secondary classes, say you specify class PRIVILEGED.USERS as primary and give it 40% of processor time. Then you specify class DEFAULT.CLASS as primary with 60%. And, you define another class (perhaps BATCH.LOW) as secondary. On this logical processor, the privileged class will get 40% (if needed); and the default class will get 60% (if needed). The secondary class will get any processing time the primaries can't use. (As mentioned above, the system will always give the primary class processes time -- even if they've used their 60% and 40% -- before giving the secondary processes time.) Processes not in the three classes won't get time on this processor.

The main point is to allot time to all classes that you want served by this processor. If class scheduling is enabled on a processor, only processes in classes that have been allotted time can get time on the processor.

2. If you want, change the time interval on which AOS/VS bases the primary percentages. For example, if you give a primary class 50% and the interval is 4.0, AOS/VS will allow the class to use up to 2 seconds of processing time (it can use more than 2 seconds if the processor would otherwise be idle).

   The default interval is 4.0 seconds -- which is suitable for most classes. Under some circumstances, described below, you might want to specify a different interval.

Allotting time involves two screens. The first screen summarizes all logical processors. It is nicknamed the "Allot summary" screen. From the "Allot summary" screen, you select a specific processor. CLASP then displays another screen, the "Allot action" screen, which you use to allot time. The BREAK/ESC key takes you from the lower level screen to the higher level one.

The default "Allot summary" screen, with only one logical processor, looks like Figure 2-6, next.

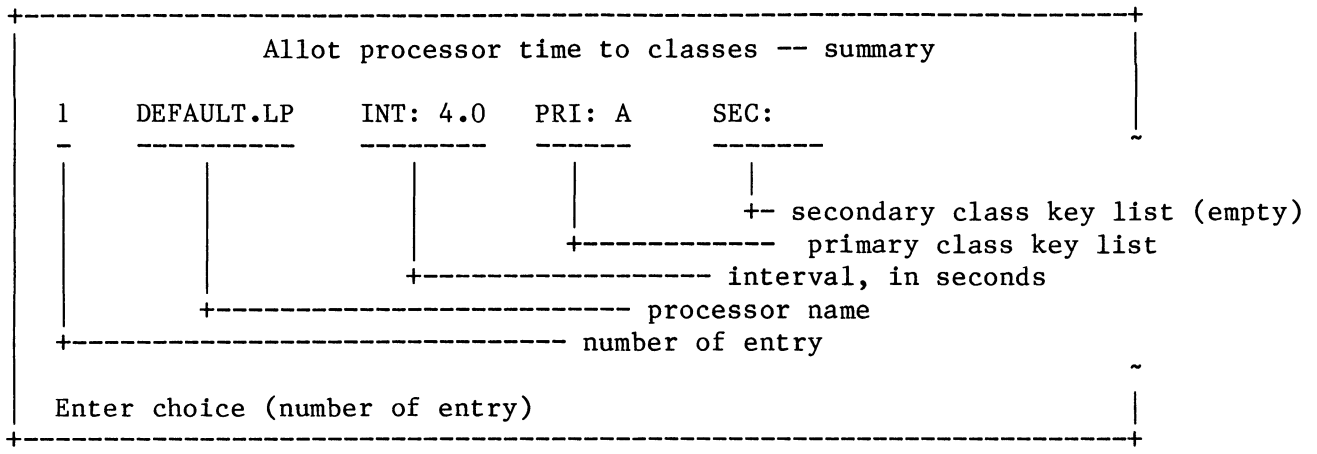

Figure 2-6. Default Allot Summary Screen

As you can see, on the default "Allot summary" screen, only the default processor (DEFAULT.LP) exists. Processor time has been allotted to the default class (name key A) only. The default class is primary. There is no secondary class. The interval is the default, 4.0 seconds.

(A class must exist (choice 1 "Create ... classes") and a logical processor must exist before you can allot the processor's time to the class. If you want a new processor to allot time on, create it: return to the Main Menu via BREAK/ESC key, select "2 Create ... processors", then return here.)

To examine or change a processor's allotment, type the number of its entry and <nl>; for example, 1 <nl> for DEFAULT.LP. CLASP will then display the "Allot action" screen, which looks like Figure 2-7:

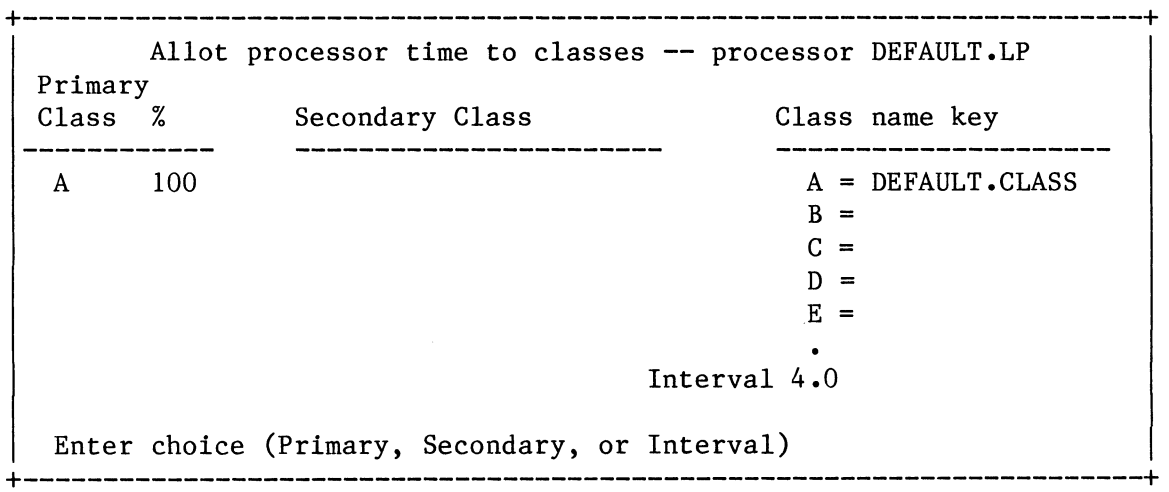

Figure 2-7. Default Allot Action Screen

As you can see from Figure 2-7, the "Allot action" screen gives more detail than the "Allot summary" screen. It shows processor DEFAULT.LP with the default class, which is primary. And it shows that the default class gets 100% of the processor's time. The interval is 4.0 seconds.

Allotting Logical Processor Time

A class can belong to the primary or secondary category. For class scheduling to work on any processor, you must specify at least one class as primary or secondary. Each category has a MODIFY command (to modify or add a class) and DELETE command (to delete a class).

For example, the following dialog shows time allotment to a class named XXX (name key B) as primary and to a class named YYY (name key C) as secondary, level 1.

```
(From Main Menu, select 3 "Allot", then choose 1 "DEFAULT.LP")
   .
(CLASP displays specific processor information)
   .
Enter choice (Primary, Secondary, or Interval):  PRIMARY <nl>  (Say primary)
Action (Modify, Delete, or ESC):  MODIFY <nl>                  (To allot, modify)
Specify class name key:  B <nl>                                (Specify class)
                      Class percentage:  40 <nl>               (Specify percent)
Action (Modify, Delete, or ESC):  BREAK/ESC                    (BREAK/ESC key
                                                               to exit)

Enter choice (Primary, Secondary, or Interval):  SECONDARY<nl>  (Say secondary)
Action (Modify, Delete, or ESC):  MODIFY <nl>                   (To allot, modify)
Specify level number:  1 <nl>                                   (Specify level)
                      Specify class name key(s):  C <nl>        (Specify a class)
                                                                For multiple
                                                                classes, use form
                                                                key, key, key)
Action (Modify, Delete, or ESC):  BREAK/ESC                     (BREAK/ESC
                                                                to exit)
Enter choice (Primary, Secondary, or Interval):
```

After this dialog, the "Allot action" screen looks like the following:

```
/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/
| Primary                                                           |
| Class  %       Secondary Class             Class name key         |
|------------    -----------------------     ---------------------- |
|  A     100%    C                             A = DEFAULT.CLASS    |
|  B     40%                                   B = XXX              |
|                                              C = YYY              |
\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\|
```

At this point, you've allotted time to both XXX and YYY classes. You might want to modify the default class (to produce a sum of 100%); then exit from the screen to the "Allot summary" screen via BREAK/ESC.

When primary class processes compete, they compete by process priority, until a class exhausts its processing percentage -- in which case, the class that exhausted its interval gets no more time until the next interval. If no primary-class process is ready to run, the secondary-class processes get their turn, level by level. Within each class, process priority works as usual -- explained in Chapter 1.)

Generally, we recommend that the sum of percentages you specify equals 100%. We suggest 100% because a sum of less than 100% may result in extra scheduling overhead, while a sum of more than 100% may allow some classes to use 100% of the time, leaving no time for other classes.

One exception to the 100% sum rule involves high priority server processes (CEO, XODIAC, INFOS II). If you create a class that includes these, the total percentage for that logical procesor can exceed 100% (For example, you might give the server class 100%, then other classes a total of 80%.) This works because these servers typically don't use a lot of processor time and thus will not use very much of the 100% -- but, if for some reason they need the time, they should be able to get

it. You'd still need to give the server processes high priority since you want them to get processor time first from among the classes that have time remaining.

The ability to define levels of hierarchy for secondary processes offers additional control for a logical processor. However, to keep things as simple as possible, you might want to keep all secondary classes on one level. Use other process controls, like priority, to control processes within secondary classes.

Changing the Time Interval

The time interval (seconds) is the period on which CLASP bases its percentage allotments. It's the number of seconds the connected job processor(s) will spend serving user processes (sum of CPU-seconds devoted to user processes). When the interval expires, or when the job processor becomes idle, the system zeros its figures (freeing each class that consumed its percentage) and starts counting again.

The default interval of 4 seconds is a good general-purpose value. This is a reasonable number for any class you think will not exhaust its percentage allotment. But in a class of interactive processes (like users in text editors) that tends to exhaust its allotment, all the class's processes may appear frozen for a second or so during each interval. Users will probably find such delays unacceptable. You can eliminate the delays by specifying a shorter interval -- say 1 or 2 seconds.

You need not change the time interval for any processor. If you want to change it, type INTERVAL <nl> in response to the "Enter choice..." prompt. CLASP will then ask "New interval:". Type the desired interval, in seconds, with fractional values in tenths of a second, if you want.

Allot Action Screen Commands

Commands you can use with this screen include the global CLASP commands and others like PRIMARY and SECONDARY. Here's a list of all commands you can use while allotting time. You can abbreviate any command.

Command	Meaning
?	Help. Lists commands.
BYE	With all screens, the BYE command exits to the CLI. Unless you've created a script or overview file (WRITE command) or applied changes (choice 6 "Apply"), any changes you made will be lost when CLASP terminates.
HELP	Help. Lists commands.
INTERVAL	Lets you change the logical processor's time interval.

Command	Meaning
PRIMARY	Lets you specify primary classes. CLASP then asks about your desired action (modify or delete). MODIFY allows you to add a class to the primary list or modify a class's allotments on the list. When you modify a class that's already on the list, CLASP will ask for a class name key and percentage. Then it will return to the "Action" prompt. To modify the next class in the primary list, use MODIFY again. (If you want to leave a class unchanged -- perhaps to change the next one -- use MODIFY and press <nl>.) When you've finished primary classes, press BREAK/ESC.
REFRESH	Refreshes the screen.
SECONDARY	Lets you specify secondary classes. CLASP then asks about your desired action (modify or delete). As with primary, secondary allows you to add a class to the secondary list or modify a level on the list. When you modify, CLASP asks about the level you want, then class name key(s) for this level. To place multiple classes on a level, type the name keys next to one another. You can, but need not, separate keys by commas; for example Specify level number: 1 <nl> Specify class name key(s): D,E,F <nl> After you identify classes on a level, CLASP returns to the "Action" prompt. To modify a different level, use MODIFY again. CLASP allows up to 16 levels of secondary classes. When you've placed classes in all the levels you want, press BREAK/ESC at the "Action" prompt. If, using choice 1 "Create ... classes", you delete all classes on a given level, CLASP will move remaining levels upward -- so that no level is empty.
WRITE	Tells CLASP to write a script or overview file. The program asks whether you want a script or overview file, then it asks for the pathname and ultimately creates the file. A script file is a list of answers to CLASP questions. It allows CLASP to run without interaction, perhaps via a CLI macro that can be started routinely by your system UP macro. An overview file summarizes classes and processors. You can print this and keep it handy near your computer, for easy reference. Creating a script file is useful whenever you want to save current CLASP settings. For a script file to impose class scheduling, you must enable class scheduling (choice 4 "Change

Command	Meaning
WRITE (continued)	processor status"). Or, to use CLASP's monitor, use choice 4 "Change processor status" to turn on accumulation mode. Script and overview files are detailed near the end of this chapter, in "Creating and Using Script and Overview Files".
BREAK/ESC key	Escapes from the current screen or question. At any prompt, this key tells CLASP to cancel the current query and return to the previous one. Pressed while you're modifying a primary or secondary class list (the "Action" prompt), BREAK/ESC leaves Modify mode and returns the "Enter choice" prompt. Use it after identifying primary or secondary classes. At "Enter choice", BREAK/ESC returns to the Main Menu, where you can select another choice (like 4 "Change processor status").

Why Use This Screen?

This screen is essential to class scheduling. For class scheduling to work, you must allot at least one logical processor's time (DEFAULT.LP's time) to at least one class. Nearly always, you'll have more than one logical processor to allot time on. And, you'll probably have more than one class to allot time to.

Also, you may want to modify existing allotments -- perhaps those in a script file, with a view toward creating an updated script file. And, you may want to change the time interval -- perhaps shorten it to minimize the effect of exhausting the interval on interactive users.

This screen allows you to allot time, change the allotment, and change the interval on a logical processor.

Generally, after allotting time or changing the time interval, the next step is to turn accumulation mode on or enable class scheduling, and/or move a job processor to this logical processor. You can do either of these things via choice 4 "Change processor status".

Example of Allotting Logical Processor Time

The default logical processor on AOS/VS startup is DEFAULT.LP. Assume that you've used CLASP to create a second logical processor, NIGHT.LP, but haven't allotted time on either processor.

Also assume you've used Main Menu choice 1 "Create...classes" to create classes. The following classes exist:

- DEFAULT.CLASS (for most interactive user processes)
- PRIVILEGED.USERS (for privileged users)
- BATCH.STD (for general-purpose batch jobs)
- BATCH.LOW (for batch jobs of low priority -- no urgency)

Having thought about these classes, you decide to allot time to them as follows, shown in Table 2-1.

Table 2-1. Processor Time and Class Plan

LP Name	Class Name	Key	Primary	Percent	Secondary	Level
DEFAULT.LP	DEFAULT.CLASS	A	yes	50	no	--
	PRIVILEGED.USERS	B	yes	40	no	--
	BATCH.STD	C	no	10	no	--
	BATCH.LOW	D	no	--	secondary	1
NIGHT.LP	PRIVILEGED.USERS	B	yes	60	no	--
	BATCH.STD	C	yes	40	no	--
	DEFAULT.CLASS	A	no	--	secondary	1
	BATCH.LOW	D	no	--	secondary	2

Listing your logical processors and classes this way (ranked by primary and secondary class) will help you allot time to them. Figure 2-17, toward the end of this chapter, includes a blank check list to help you rank them.

Having planned your logical processors and classes, and created the classes and logical processor, you can proceed to allot processor time. From CLASP's Main Menu, take choice 3 "Allot". The "Allot summary" screen looks like Figure 2-8:

```
                Allot processor time to classes -- summary

  1    DEFAULT.LP     INT:  4.0     PRI: A      SEC:
  2    NIGHT.LP       INT:  4.0     PRI: A      SEC:

  Enter choice (number of entry)
```

Figure 2-8. Allot Summary Screen

The "Allot summary" screen shows two logical processors, DEFAULT.LP and NIGHT.LP. Since time has not been allotted, DEFAULT.LP shows its

original settings: it has the class whose name key is A defined as primary, it has no secondary classes, and its time interval remains 4 seconds. Processor NIGHT.LP has just been created, thus has the same settings as the original default, DEFAULT.LP.

Using the information from Table 2-1, you might allot processor time as shown in multiple-page Figure 2-9. This figure follows.

```
Console Dialog                                          Explanation of Dialog
--------------                                          ---------------------

.                                                       (Main Menu display)
Enter choice  3 <nl>                                    (Select "Allot";
                                                        CLASP displays "Allot
+-------------------------------------------+           summary" screen)
|        Allot processor time to classes ...|
|                                           |
| 1  DEFAULT.LP     INT: 4.0    PRI: A    SEC:
| 2  NIGHT.LP       INT: 4.0    PRI: A    SEC:
|                                           |
|Enter choice  1 <nl>                       |           (Select DEFAULT.LP)
+-------------------------------------------+           (CLASP displays "Allot
                                                        action" screen)
+-----------------------------------------------------------------+
|           Allot processor time to classes ...  DEFAULT.LP       |
|                                                                 |
| Primary                                                         |
| Class  %          Secondary Class              Class name key   |
| -------------     ----------------------       ---------------- |
|  A     100                                     A = DEFAULT.CLASS|
|                                                B = PRIVILEGED.USERS
|                                                C = BATCH.STD    |
|                                                D = BATCH.LOW    |
|                                                E =              |
|                                                F =              |
|                                                .                |
+-----------------------------------------------------------------+

Enter choice (Primary, Secondary...  PRIMARY <nl>       (Specify primary)
Action (Modify, Delete ...  MODIFY <nl>                 (Modify a class)
Specify class name key:  A <nl>                         (A for default class)
                    Class percentage:  50 <nl>          (Give class 50%)
Action (Modify, Delete ...  MODIFY <nl>                 (Modify another class)
Specify class name key:  B <nl>                         (B for privileged class)
                    Class percentage:  40 <nl>          (Give class 40%)
Action (Modify, Delete ...  MODIFY <nl>                 (Modify another class)
Specify class name key:  C <nl>                         (C for batch standard)
                    Class percentage:  10 <nl>          (Give class 10%)
Action (Modify, Delete ...  BREAK/ESC                   (BREAK/ESC key ends
                                                        modify operation)
Enter choice (Primary, Secondary...  SECONDARY <nl>     (Specify secondary)
Action (Modify, Delete ...  MODIFY <nl>                 (Modify a class)
Specify level number:  1 <nl>                           (Choose level 1)
                    Class name key(s):  D <nl>          (Put low level batch
                                                        jobs on level 1)
Action (Modify, Delete ...  BREAK/ESC                   (BREAK/ESC key ends
                                                        modify operation)
Enter choice (Primary, Secondary, or Interval)          ("Allot action" screen
                                                        prompt returns)
```

Figure 2-9. Allotting Classes Time on Two Logical Processors (continues)

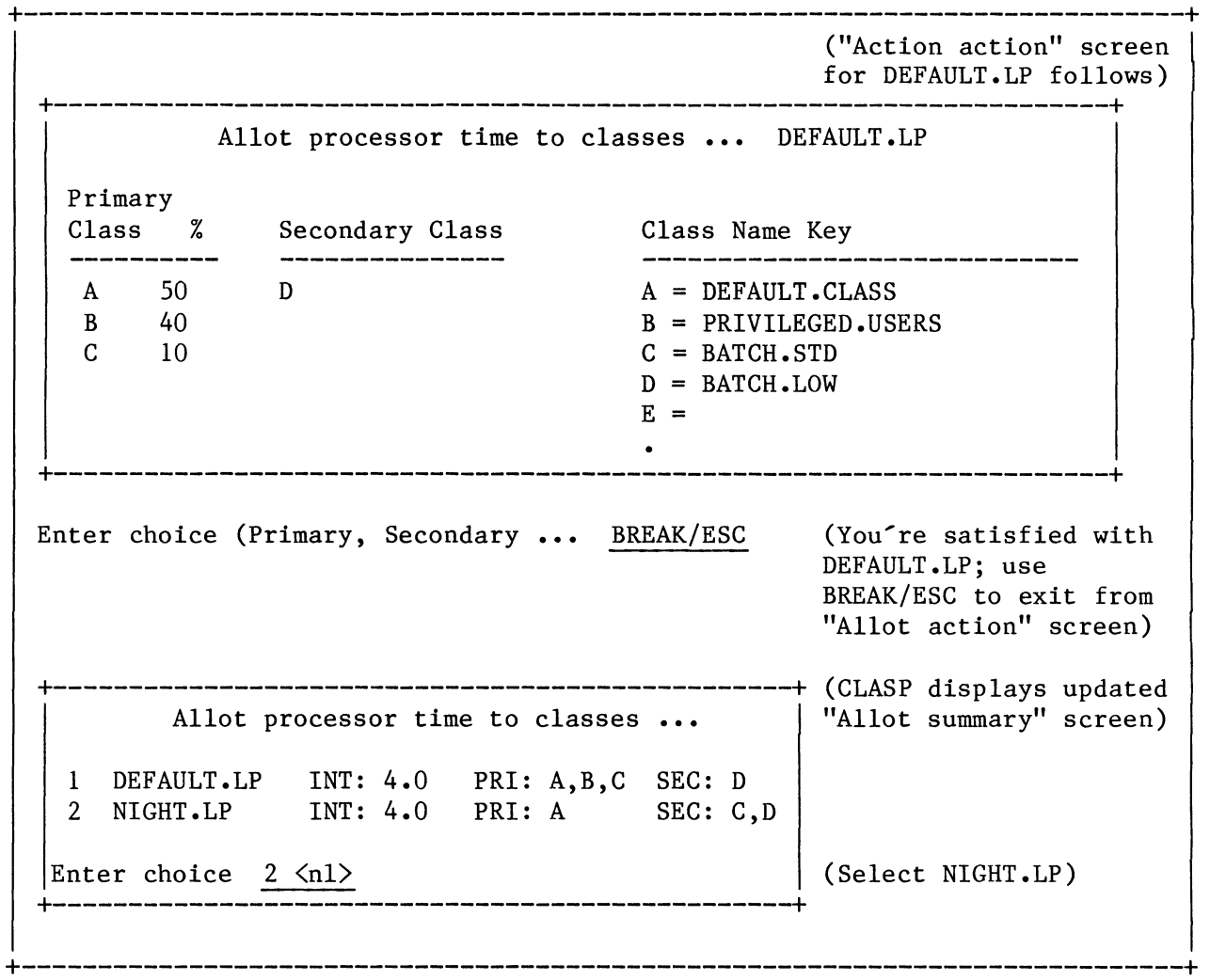

Figure 2-9. Allotting Classes Time on Two Logical Processors (continued)

```
                                                    (CLASP displays "Allot
                                                    action" screen, NIGHT.LP)
+-------------------------------------------------------------------+
|          Allot processor time to classes ...  NIGHT.LP            |
|                                                                   |
| Primary                                                           |
| Class   %     Secondary Class        Class Name Key               |
| -------   ---------------        ----------------------------     |
|  A    100                        A = DEFAULT.CLASS                |
|                                  B = PRIVILEGED.USERS             |
|                                  C = BATCH.STD                    |
|                                  D = BATCH.LOW                    |
|                                  E =                              |
|                                  .                                |
+-------------------------------------------------------------------+

Enter choice (Primary, Secondary ...  PRIMARY <nl>   (Specify primary)
Action (Modify, Delete ...  MODIFY <nl>              (Modify a class)
Specify class name key:  B <nl>                      (B for privileged class)
                      Class percentage:  60 <nl>     (Give class 60%)
Action (Modify, Delete ...  MODIFY <nl>              (Modify another class)
Specify class name key:  C <nl>                      (C for batch jobs, std)
                      Class percentage:  40 <nl>     (Give class 40%)
Action (Modify, Delete ...  BREAK/ESC                (BREAK/ESC key ends
                                                     primary operation)
Enter choice (Primary, Secondary ...  SEC <nl>       (Specify secondary)
Action (Modify, Delete ...  MODIFY <nl>              (Modify a class)
Specify level number:  1 <nl>                        (Secondary level 1)
                      Class name key(s):  A <nl>     (A for default class)
Action (Modify, Delete ...  MODIFY <nl>              (Modify another class)
Specify level number:  2 <nl>                        (Secondary level 2)
                      Class name key(s):  D <nl>     (D for batch jobs, low)
Action (Modify, Delete ...  BREAK/ESC                (BREAK/ESC key ends
                                                     modify operation)
Enter choice (Primary, Secondary ...
                                                     ("Allot action" screen
                                                     for NIGHT.LP follows)
+-------------------------------------------------------------------+
|          Allot processor time to classes ... NIGHT.LP             |
|                                                                   |
| Primary                                                           |
| Class   %     Secondary Class        Class Name Key               |
| -------   ---------------        ----------------------------     |
|  B    60      A                    A = DEFAULT.CLASS              |
|  C    40      D                    B = PRIVILEGED.USERS           |
|                                    C = BATCH.STD                  |
|                                    D = BATCH.LOW                  |
|                                    E =                            |
|                                    .                              |
+-------------------------------------------------------------------+
```

Figure 2-9. Allotting Classes Time on Two Logical Processors (continued)

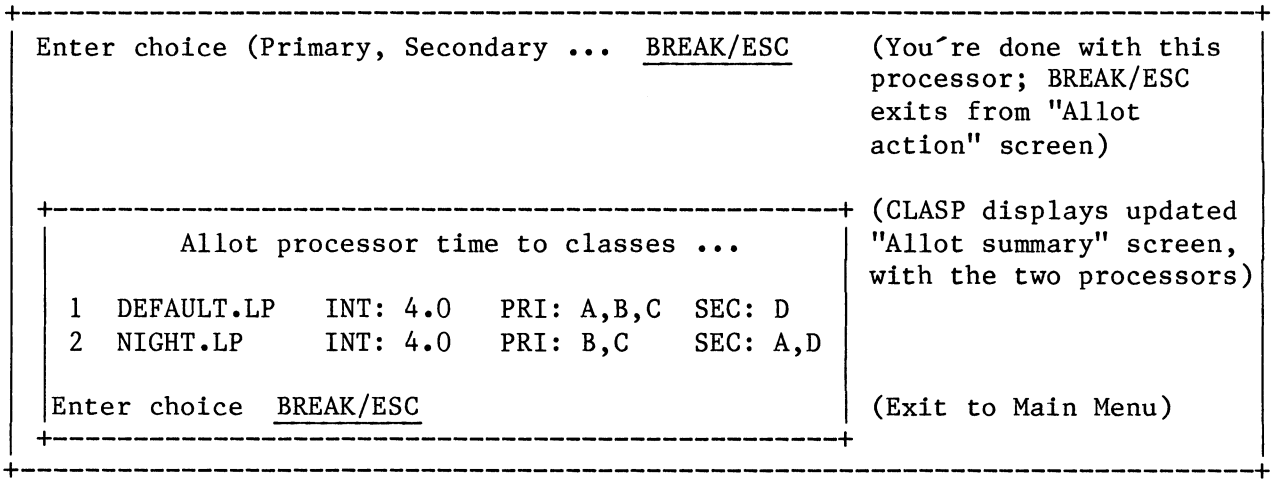

Figure 2-9. Allotting Classes Time on Two Logical Processors (concluded)

Change Processor Status -- Choice 4

Choice 4 lets you

- Enable or disable class scheduling, or activate accumulation mode (to monitor a logical processor). You must use enable if you want class scheduling to occur. Accumulation mode is very useful for class monitoring.

- Move a job processor (physical processor) to a logical processor other than the default processor (DEFAULT.LP). When you initialize any job processor, AOS/VS automatically connects it to DEFAULT.LP.

Enabling Class Scheduling

When AOS/VS starts up, class scheduling is disabled on all logical processors, including the default logical processor.

Until CLASP runs and (via a script file or interactively) enables class scheduling on a logical processor, AOS/VS enforces standard scheduling: when a job processor is free, the highest priority, ready process gets control of it.

Thus, if you want to use classes, enabling is an essential step. You can enable scheduling on a logical processor at any time -- before or after moving a job processor to or from the processor or alloting time on it.

For class scheduling to operate, all the following must be true:

- At least one nondefault user or program locality must have been defined with the PREDITOR or SPRED programs. (Without this, all processes will belong to the same class, DEFAULT.CLASS, and CLASP settings will have no effect.)

- One or more classes must have been created and/or reassigned a locality pair, via CLASP Main Menu choice 1 "Create ... classes".

- On a logical processor, time must have been allotted to one or more classes, via the CLASP Main Menu choice 3 "Allot". The logical processor can be the default; or it can be a processor you've created with the CLASP Main Menu choice 2 "Create ... processors".

- On a logical processor, class scheduling must be enabled. If this logical processor is not the default, then a job processor must be moved to it. Both the enable and the move can occur via the CLASP Main Menu choice 4 "Change processor status".

- CLASP settings must have been applied to the system. You can do this interactively via the CLASP Main Menu choice 6 "Apply". Noninteractively, you can do it by specifying a script file and the /BATCH switch when you start CLASP.

You can fulfill any of these previous requirements at any point, whether or not class scheduling has been enabled. (Generally, though, you'll

want to use CLASP to set up the environment -- including enabling class scheduling -- before writing a script file.)

CLASP shows that class scheduling has been enabled by displaying an E in the logical processor's State column.

Disabling Class Scheduling

When you disable class scheduling on a logical processor, the system stops recognizing classes on the processor: standard AOS/VS scheduling, by process priority only, returns for this processor. Use disable when you want to resume standard scheduling. Disabling is not required before moving a job processor to or from a logical processor.

CLASP shows that class scheduling has been disabled by displaying a D in the logical processor's State column.

Accumulating Class Use Information

Accumulation mode is a logical processor state, like enabled and disabled. In accumulation mode, CLASP gathers information on class use without enabling class scheduling. Accumulation mode is designed to help you monitor class use and refine class allotment percentages. To see accumulated information, use Main Menu choice 7 "Monitor".

If you tell CLASP to accumulate, CLASP will disable class scheduling (if enabled) and turn on accumulation mode.

CLASP shows that it's accumulating information by displaying an A in a logical processor's State column.

Moving a Job Processor

The MOVE command allows you to connect a job processor to a logical processor -- activating the logical processor or increasing its power.

At startup, AOS/VS runs on only one job processor: the mother processor, by default job processor 0. By default, job processor 0 is connected to the default logical processor, DEFAULT.LP.

After startup, if your computer has multiple job processors, someone or something (perhaps the UP macro) should initialize the second and any other job processors using the CLI command JPINITIALIZE. The JPINITIALIZE command tells AOS/VS to recognize these job processors -- and connect them to the default logical processor, DEFAULT.LP.

Then, this screen's MOVE command lets you move a job processor to a logical processor other than the default. Generally, you'll use MOVE after creating classes and before applying settings, as follows.

- Define classes and logical processors, allot classes time
- Establish desired state (enable or accumulation mode)
- Move one or more job processors
- Apply settings (interactively or via script file)

You can enable class scheduling or turn on accumulation mode before or after moving a job processor to a logical processor.

When you specify a move, CLASP asks you to identify the job processor and logical processor. After you do so, CLASP returns the "Enter choice" prompt.

Moving a job processor can give you a brand-new scheduling environment, with entirely different class scheduling. But there are some cautions to observe -- explained next.

Moving a Job Processor from an Active Logical Processor

If a logical processor has only one job processor, and you move the job processor to another logical processor, this leaves the original logical processor without a job processor. The original logical processor (which is simply a scheduling definition) becomes inactive. If a class with time allotted on the original logical processor also has time allotted on an active logical processor, AOS/VS will schedule this class on the active logical processor.

But, if a class with time on the original logical processor has no time on an active logical processor, the processes in that class become stranded. They won't be scheduled, and are frozen until a processor with time allotted to their class becomes active.

The following scenario shows how a process can be stranded this way.

1. AOS/VS comes up as usual, with job processor 0 connected to DEFAULT.LP.

2. With CLASP, you create class XXX, user locality 15 and program locality 0. Then you create a logical processor MYPROC. You allot XXX time on DEFAULT.LP but don't allot it time on MYPROC.

3. A user process comes up -- no matter how -- in user locality 15 and program locality 0. This means it runs in class XXX.

4. With CLASP -- perhaps via a script file -- class scheduling on both processors is enabled. And, job processor 0 is moved to logical processor MYPROC. This deprives DEFAULT.LP of its job processor.

   Since class XXX isn't allotted time on any other processor, its processes can't be scheduled. The user process running in class XXX

becomes stranded. The process appears frozen. It doesn't die, but it won't run again until class scheduling is disabled on MYPROC, or a job processor is moved to an logical processor where class XXX can run (as, for example, if you moved job processor 0 back to DEFAULT.LP, or if you allotted class XXX time on MYPROC).

A stranded process is treated the same way as a blocked process. AOS/VS maintains the process's state as long as AOS/VS runs. PED displays the process's status as active (not blocked -- B -- or swapped -- S). At system shutdown, the process will be terminated along with all others.

If your site will move job processors, you'll want to look for stranded processes as you check processes with the ? macro before shutdown. With the /CLASSNAME switch, PED will tell you class names. If you determine that a process has been stranded, and you want it to run to completion and/or shut down normally, you can use CLASP to give its class some processor time. To do this, use Main Menu choice 3 "Allot" or use this "Change processor status" screen to move a job processor back to the logical processor. Or, you can disable class scheduling on all processors, which will let the process compete for time.

The ability to connect job processors to other logical processors can be very useful. If you use it extensively, be aware of the impact of depriving a logical processor of a job processor -- and, especially, of the possibility of stranding a process.

The Change Processor Status Screen

A "Change processor status" screen -- on an MV/20000 Model 2 system with four logical processors defined -- looks like Figure 2-10.

```
+-------------------------------------------------------------------+
|                                                                   |
|                      Change processor status                      |
|                                                                   |
|   Logical Processors      State   Job Processors                  |
|   ------------------      -----   --------------                  |
|                                                                   |
|   1  DEFAULT.LP             E     0                               |
|   2  NIGHT.LP               A     1                               |
|   3  FAVORS.BATCH           D                                     |
|   4  RACING.SORTS           E                                     |
|                                                                   |
|                                                                   |
|   Enter choice (Accumulate, Disable, Enable, or Move)             |
|                                                                   |
+-------------------------------------------------------------------+
```

Figure 2-10. Change Processor Status Screen

Change Processor Status Commands

Commands you can use with this screen include the global CLASP commands and others like ENABLE and MOVE. Here's a list of all commands you can use while changing status. You can abbreviate any command.

Command	Meaning
?	Help. Lists commands.
ACCUMULATE	Tells the system to accumulate data for monitoring class use without enforcing class scheduling. As a prelude to accumulating data, CLASP disables class scheduling (if enabled). Use this command when you're interested in monitoring class use on a logical processor, via Main Menu choice 7 "Monitor". After you type the ACCUMULATE command, CLASP asks Specify logical processor menu number or A for all LPs: Type the number of the logical processor you want the system to accumulate class data on, or type A <nl> to accumulate data on all logical processors. The "Enter choice" prompt returns. Accumulation mode involves a slight -- but real -- amount of system overhead, so you shouldn't leave it on for prolonged periods unless you're monitoring. It's intended to help you find useful settings to build into a script file. In the same script file, you'd specify "Enable", not "Accumulate" mode. While accumulating, CLASP displays an A in the processor's State column. The processor will continue accumulating after you leave CLASP -- unless you select another state, like Enable, from CLASP.
BYE	With all screens, the BYE command exits to the CLI. Unless you've created a script or overview file (WRITE command) or applied changes (choice 6 "Apply"), any changes you made will be lost when CLASP terminates.

Command	Meaning
DISABLE	Disables class scheduling on one or all logical processors. Scheduling reverts exclusively to AOS/VS's standard arrangement: when a job processor is free, the highest priority, ready process gets control of it. Disabling is useful when you want to restore standard scheduling or turn off accumulation mode). After you type the DISABLE command, CLASP asks Specify logical processor menu number or A for all LPs: Type the number of the logical processor to disable, or type A <nl> to disable on all processors. The "Enter choice" prompt returns. While a processor is disabled, CLASP displays a D in its State column.
ENABLE	Enables class scheduling on one or all logical processors. AOS/VS then schedules processes according to each enabled processor's class definitions. By default, all logical processors, including the default, have class scheduling disabled. AOS/VS won't use class scheduling unless it's enabled on at least one logical processor. (To accumulate statistics for the monitor, use ACCUMULATE instead of ENABLE.) After you type the ENABLE command, CLASP asks Specify logical processor menu number or A for all LPs: Type the number of the logical processor to enable, or type A <nl> to enable all processors. The "Enter choice" prompt returns. While a processor is enabled, CLASP displays an E in its State column.
HELP	Help. Lists commands.
MOVE	Moves a job processor to a logical processor. This command allows you to activate a different logical processor, or -- with multiple job processors -- add another job processor to a logical processor. The source and target logical processors can have class scheduling enabled or disabled; or they can be accumulating monitor information. When you type the MOVE command, CLASP asks about the job and logical processors. Then, CLASP validates your answers and updates the screen display. The "Enter choice" prompt returns.

Command	Meaning
MOVE (continued)	CLASP doesn't actually move the job processor until you apply changes (Main Menu choice 6 "Apply") or start CLASP with /BATCH and a script file that specifies a move. A sample MOVE dialog, that moves job processor 0 to logical processor NIGHT.LP, is Enter choice (Accumulate, Disable, Enable, or Move): MOVE <nl> Specify job processor number: 0 <nl> Specify logical processor entry number: 2 <nl> Enter choice (Accumulate, Disable, Enable, or Move): There's a potential pitfall in the MOVE command: it's possible to strand processes, as explained earlier in "Moving a Job Processor from an Active Logical Processor".
REFRESH	Refreshes the screen.
WRITE	Tells CLASP to write a script or overview file. The program asks whether you want a script or overview file; then it asks for the pathname and ultimately creates the file.
BREAK/ESC key	Escapes from the current screen or question. At any prompt, this key tells CLASP to cancel the current query and return to the previous one. At "Enter choice", it returns to the Main Menu, where you can select another choice (like 6 "Apply" or 7 "Monitor").

When you're running CLASP interactively, the act of turning on accumulation mode, enabling, disabling, or moving doesn't occur until you apply changes -- via Main Menu choice 6 "Apply". Noninteractively, the act(s) occurs when you run CLASP, specify a script file that contains different State settings, and include the /BATCH switch.

If you want to continue interactively, and are done with this "Change processor status" screen, press the BREAK/ESC key to return to the Main Menu, then continue from there.

If you want to create a script file, and you're satisfied with the environment settings in CLASP, you can type WRITE <nl>, specify the script name, then exit from CLASP and test the script file by typing X CLASP/BATCH/SCRIPT_FILE=pathname.

To be active, a logical processor must have at least one job processor attached. From the class viewpoint, active means having scheduling enabled, one or more job processors attached, and CLASP settings applied. But, you can define different logical processors, and, via a script file, easily move a job processor from one logical processor to another.

Why Use This Screen?

This screen is essential to class scheduling: for class scheduling to work, you must enable it on at least one logical processor.

Also, a primary benefit of class scheduling is the ability to change logical processors easily. This screen's MOVE command can give you a brand-new scheduling environment, with entirely different class scheduling.

And, this screen lets you tell the system to accumulate statistics. The monitor screen can then help you correct and refine your class percentage allotments.

Generally, after using this screen, the next step is to apply changes. Do this via Main Menu choice 6 "Apply" or create a script file, exit from CLASP, and test the script via X CLASP/BATCH/SCRIPT_FILE=pathname.

Example of Changing Processor Status

Figure 2-11 shows a change processor status dialog that enables two logical processors, then moves job processor 0 to the second logical processor, NIGHT.LP.

```
Console Dialog                                    Explanation of Dialog
----------------------------------------------    ---------------------

) X CLASP <nl>                                    (Start CLASP)
.                                                 (It displays Main Menu)

.                                                 (Create classes, create
.                                                 logical processor
.                                                 NIGHT.LP, allot time on
                                                  NIGHT.LP, and return to
                                                  Main Menu)
Enter choice 4 <nl>                               (Select "Change
                                                  processor status"
+-------------------------------------------+     screen; CLASP displays
|          Change processor status          |     the screen)
|                                           |
|  Logical Processor    State   Job Processors
|  -----------------    -----   --------------
|                                           |
|  1  DEFAULT.LP        D       0           |
|  2  NIGHT.LP          D                   |
|                                           |
|  Enter choice (Accumulate, Disable,       |
|              Enable, or Move) ENABLE <nl> |     (Specify ENABLE)
+-------------------------------------------+

 Specify logical processor menu number... 1 <nl>  (Enable on DEFAULT.LP)

 Enter choice (Accumulate, Disable,
              Enable, or Move) ENABLE <nl>        (Want to enable another)
 Specify logical processor menu number... 2 <nl>  (Enable on NIGHT.LP)

 Enter choice (Accumulate, Disable,
            Enable, or Move): MOVE <nl>           (Specify move)
 Specify job processor number: 0 <nl>             (Move job processor 0...
 Specify logical processor menu number... 2 <nl>  ... to NIGHT.LP)

                                                  ("Change processor
+-------------------------------------------+     status" screen looks
|          Change processor status          |     like this)
|                                           |
|  Logical Processor    State   Job Processors
|  -----------------    -----   --------------
|                                           |
|  1  DEFAULT.LP        E                   |
|  2  NIGHT.LP          E       0           |
|                                           |
|  Enter choice (Accumulate, Disable,       |
|              Enable, or Move): BREAK/ESC  |     (Done with screen; exit)
+-------------------------------------------+     (Main Menu returns;
                                                  select "Apply" choice)
```

Figure 2-11. Change Processor Status Example

Display Processor Hardware Information -- Choice 5

Choice 5 displays job processor hardware information, including

- job processor number;
- job processor type (mother or child);
- job processor microcode revision;
- job processor ID (CPUID); and
- whether the job processor has a hardware floating-point unit.

This screen accepts all CLASP global commands: ?, HELP, REFRESH, WRITE, and BREAK/ESC.

The job processor ID (CPUID) identifies the computer name and certain other options (explained in Table 2-2).

Table 2-2. DG Computer CPU Identifiers (CPUIDs) and Names

CPU ID	Computer Name
8468	MV/4000 with independent hardware IOC board
8469	MV/4000 with independent hardware IOC board and hardware floating-point unit
8760	MV/4000
8761	MV/4000 with hardware floating-point unit
8764	MV/4000 with graphics instruction set (GIS)
8765	MV/4000 with graphics instruction set (GIS) and hardware floating-point unit
8780	MV/10000
8880	MV/10000 SX
8930	DS/7700
8936	DS/7500
8936	MV/2000 DC
8998	MV/20000
8999	MV/20000 with hardware floating-point unit
9340	MV/8000
9341	MV/8000 with hardware floating-point unit
9670	MV/8000 C
9671	MV/8000 C with hardware floating-point unit
9680	MV/8000 II
9681	MV/8000 II with hardware floating-point unit
9690	MV/6000

Why Use This Screen?

On a system with multiple job processors, this screen is useful when you want hardware information -- perhaps when you plan to move a job processor.

Display Hardware Information Example

An example display follows in Figure 2-12.

```
+-------------------------------------------------------------------------+
| Console Dialog                                  Comments on Dialog      |
| ----------------------------------------------  ----------------------  |
|                                                                         |
| .                                               (Main Menu display)     |
| Enter choice  5 <nl>                            (Select "Processor      |
|                                                 hardware information")  |
|                                                                         |
|                                                 (CLASP displays the     |
|                                                 screen)                 |
|  +-----------------------------------------------------+                |
|  |       Display processor hardware information        |                |
|  |                                                     |                |
|  | Job              Microcode              Hardware    |                |
|  | Proc    Type     revision     CPU ID      FPU       |                |
|  | ----    ------   ---------    ------   --------     |                |
|  |  0      Mother      6.00       9681      Yes        |                |
|  |                                                     |                |
|  |                                                     |                |
|  | Enter choice  BREAK/ESC                             |  (Exit to Main |
|  +-----------------------------------------------------+   Menu)        |
|                                                                         |
+-------------------------------------------------------------------------+
```

Figure 2-12. Displaying Processor Hardware Information

Apply CLASP Settings to the Operating System -- Choice 6

Choice 6 tells CLASP to issue system calls that apply all current CLASP environment settings to the operating system. These include

- Class creations and locality pair definitions (1 "Create ... classes") including updates of class name identifier files in :PER

- Logical processor creations (2 "Create ... processors"), including updates of processor name identifier files in :PER

- Processor time allotments to classes (3 "Allot")

- Class state changes on logical processors (like enabling scheduling or turning on accumulation mode (4 "Change processor status")

- Job processor moves to logical processors (4 "Change processor status")

CLASP settings are not applied to AOS/VS -- when you run CLASP interactively -- until you select this "Apply" choice. So, unless you're just viewing (/VIEW_ONLY switch), experimenting, or creating a script/overview file, you should use this choice before leaving CLASP.

If you want to create a script file, you need not select this choice -- since, after you create the script file and leave CLASP, you can apply the settings via X CLASP/BATCH/SCRIPT_FILE=pathname. But you can select this choice to test CLASP settings if you want.

If you want to create a script file, and you're satisfied with the environment specified in CLASP, you can type WRITE <nl> and proceed as described in the next section.

Whether or not you're creating a script file, you must have enabled class scheduling (4 "Change processor status") for class scheduling to work. For most useful monitoring, you must have turned on accumulation mode (7 "Monitor").

Applying settings includes several different steps, including system calls and file I/O in directory :PER. It may take some time -- say 10 to 20 seconds -- to complete.

Why Use This Choice?

As mentioned above, when you're running CLASP interactively, you must select "Apply" to apply CLASP settings to AOS/VS. Thus, if you want to implement, monitor, or test the new settings interactively, you must select this choice.

Also, certain errors can't be detected until CLASP settings are applied to the system. Applying settings interactively is a good way to check for these errors. The messages are explained in Chapter 4.

Monitor Logical Processors -- Choice 7

Choice 7 allows you to monitor (examine) class scheduling on one or more logical processors. This choice can help you decide on the right class time allotment before you create a script file.

For monitoring to work, one or more classes must exist (Main Menu choice 1 "Create ... classes"); processor time must be allotted to classes (choice 3 "Allot"), and the logical processor must have a job processor connected to it (choice 4 "Change processor status"). Also, CLASP settings must have been applied to the operating system (via an earlier CLASP run or in the current CLASP process via Main Menu choice 6 "Apply).

You can monitor a logical processor while it's in any of the three states: disabled, accumulating, or enabled. In any state, CLASP reports how connected job processor(s) time is being spent: on behalf of users, on behalf of the system, or in the idle loop. This job processor time breakdown can tell you whether the logical processor is busy enough to warrant monitoring. In addition to user, system, and idle figures, class information is displayed as follows.

Disabled — Monitoring while class scheduling is disabled gives no percentage information on class use.

Accumulating — Monitoring while a processor is accumulating shows how much time classes are getting when class scheduling is not enforced. Monitoring while accumulating provides useful feedback on a class's actual percentage use. This feedback can help you correct and refine percentage allotments and other class definitions.

Enabled — Monitoring while class scheduling is enabled shows how much time classes are getting while class scheduling is enforced. This shows how faithfully the system is enforcing your allotments. Each primary class should get about the percentage specified for it. If not, you may have specified a class percentage sum of more than 100% -- or, there may be too little load (too much idle time) for the system to impose class scheduling.

You can change the state to accumulating, enabled, or disabled via screen 4 "Change processor status".

The state displayed is based on CLASP's record of status, not on actual AOS/VS status. The two may differ if you haven't applied settings. For example, if class scheduling is disabled and you turn accumulation mode on, CLASP will report State A -- but accumulation mode won't really be on until you apply settings via 6 "Apply".

Like other CLASP choices, the monitor has a summary screen and a specific processor screen. The summary screen lets you select the logical processor to monitor; the processor screen gives the information. Both screens give processor state information.

When you type 7 <nl> from the Main Menu, CLASP displays the "Monitor summary" screen. The original monitor summary screen, with only the default processor defined, looks like Figure 2-13, next.

```
+-------------------------------------------------------------------+
|                  Monitor logical processors -- summary            |
|                                                                   |
|     Logical Processor     State                                   |
|     -----------------     -----                                   |
|                                                                   |
|     1  DEFAULT.LP         D                                       |
|                                                                   |
|  Enter choice (number of entry)                                   |
+-------------------------------------------------------------------+
```

Figure 2-13. Original Monitor Summary Screen

After you specify a logical processor by number, CLASP displays the "Monitor processor" screen. With accumulation mode on, a "Monitor processor" screen for the default logical processor might look like Figure 2-14, next.

```
+-------------------------------------------------------------------------+
| Nov 18, 1985   Monitor processor: DEFAULT.LP        State: A     4:44:04 PM |
|-------------------------------------------------------------------------|
|                                                                         |
| Primary          Percent used since      CPU usage: CLASP start/last cycle |
| Classes  Percent CLASP start/last cycle  ---------------------------------|
| -------  ------- ----------------------     User          76%          78% |
|   A       55%       45%          47%       System        23%          22% |
|   B       45%       23%          21%       Idle           1%           0% |
|                                                                         |
|                                                                         |
|                                             Class name key              |
|                                             --------------              |
|                                             A = DEFAULT.CLASS           |
| Secondary        Percent used since         B = PRIVILEGED.USERS        |
| Classes   Level  CLASP start/last cycle     C = LOW_PRIORITY            |
| ---------  -----  ----------------------     D =                         |
|    C         1        34%          32%      E =                         |
|                                             .                           |
+-------------------------------------------------------------------------+
```

Figure 2-14. Example Monitor Processor Screen

As you can see, CLASP displays two sets of class use figures: the average percentage used since CLASP start and during the last CLASP cycle. (Default cycle time is 10 seconds; you can subtact one second by pressing < and add one second by pressing >.)

In a heavily loaded system, the CLASP cycle may lag (as CLASP competes with other processes for processor time). You can improve CLASP response time by creating the CLASP process as a preemptible type (via a command like PROCESS/TYPE=PREMPTIBLE/IOC/BLOCK CLASP <nl>. For this to work, everyone who uses this command must have "Change type" privilege in his or her user profile.

If CLASP shows a class using 0%, the class may actually be getting no time. Or, CLASP settings that turn on accumulation mode (or enable class scheduling) haven't been applied. If you see a figure of 0%, review your steps to make sure settings have been applied. If you can't tell whether they've been applied, return to the Main Menu and apply them (choice 6); then return here.

If class scheduling is enabled, settings have been applied, your computer has more than one job processor, and CLASP shows a figure that differs significantly from allotted (Percent) figures, the system may be ignoring allotted figures in the interest of efficiency. AOS/VS ignores allotted figures if enforcing them would leave a job processor idle.

Here's an example. Say you have an MV/20000 Model 2 system (it has two job processors). You run one logical processor with several classes, including a class named FP. Class FP usually has only one process, that does continuous computations. You allotted 60% of processor time to class FP and 40% to all other classes. CLASP's monitor shows class FP with about 50% and other classes with about 50%. This happens because a process can run in only one processor at a time. The FP-class process runs continuously in a processor. Processes in other classes run in the other processor until they consume the 40% allotment. At that point, the system looks for another another eligible-class process to run, finds none, and zeros its figures, freeing the other-class processes to run again. Again they run until they've consumed 40%; again the system looks for processes to run, finds none, and zeros its figures. The FP-class process continues running thoughout. As this sequence repeats, each set (FP class and other classes) comes to average about 50%, not 60%-40% as allotted.

This situation occurs with multiple job processors when, to satisfy allotted percentages -- there aren't enough processes in a class (as with FP) to keep all job processors active for the specified time. When the situation occurs, the only way AOS/VS could satisfy your percentages would be to keep a processor idle, which would be inefficient. If the situation does occur, one way to compensate is to make the favored process (above, the one in class FP) into two or more processes, allowing the system to run processes from the same class in two job processors simultaneously.

Monitor Processor Screen Commands

Commands you can use with this screen include REFRESH (to refresh the screen display) and ESC to return to the monitor summary screen.

CLASP updates the "Monitor processor" display every 10 seconds. You can reduce this cycle update time or increase it by pressing the left or right angle bracket key (< or >). To reduce cycle time by 1 second, press < and to increase cycle time by one second, press >.

If you want to create a script file, and you're satisfied with the environment settings in CLASP, make sure each logical processor has the state you want (for example, using choice 4 "Change processor status"). Then, type WRITE <nl>, specify the script file pathname, then exit from CLASP and test the script file by typing X CLASP/BATCH/SCRIPT=pathname.

Why Use This Screen?

The "Monitor summary" screen offers two important benefits: it can indicate whether classes are worthwhile for your current environment, and it can give invaluable feedback on your class definitions.

The first enefit involves the "Idle" figure. If -- in accumulation mode for the default logical processor -- the screen shows an idle time of over 3%, this means you have more than enough job processor power for current demands.

A high idle figure (more than 3%) may mean that you're not running under typical load (running all the processes your application usually runs). If you're not running under typical load, arrange to do so; monitoring your system is not useful unless it's performing under load. If you are running under typical load and see a high idle figure for the default logical processor, this may mean that you don't need classes and logical processors at all. Classes and logical processors can improve throughput only when all job processor time is being consumed; if it's not being consumed, classes and LPs are irrelevant.

The second benefit is feedback on your class definitions. Class scheduling is not simple: it involves an understanding of process priority, your user environment, and interrelationships between PREDITOR- and SPRED-specified parameters. Feedback on class definitions, especially in accumulation mode, can be quite valuable.

After creating classes, you can allot dummy percentages (say 100%) to each class, then accumulate information and use the monitor to see what percentages each class normally gets. Then, you can return to the "Allot action" screen and adjust class allotments according to the monitor screen figures.

After using the monitor screen and tailoring allotments, you may want to enable scheduling and create a script file.

Monitor Example

This example assumes your goal is to let processes that run in batch get more job processor time. To make this happen, you'll put batch jobs in their own class. Since the class will be program oriented (not user oriented), you'll have the program locality establish the class. Say that the class will include all user processes that run a program with a program locality of 2. You run SPRED on the pertinent program file(s) and specify a program locality of 2.

Using CLASP, you create the class (you name it BATCH.STD) and reassign locality pairs to match your plan. Program locality 2 will determine the class -- so you give BATCH.STD a program locality. And, you give it all user localities (respond <nl> to CLASP's "User locality" question).

Next, you allot time on your general-purpose logical processor (DEFAULT.LP) to class BATCH.STD. Say you make BATCH.STD a primary class and give it 100%. (To be scheduled, thus to be measured, a class must

have been allotted time on a logical processor. The allotment percentage need not be precise, since class scheduling isn't enforced in accumulation mode. So, you can give each class 100%.) You check that a job processor is connected to your logical processor as usual. Then you make sure your normal applications environment is running.

Finally, you select choice 7 "Monitor", select your active logical processor, and see a display like the following (Figure 2-15).

```
+-------------------------------------------------------------------------+
| Nov 18, 1985   Monitor processor: DEFAULT.LP        State: A     4:46:11 PM |
|-------------------------------------------------------------------------|
|                                                                         |
| Primary          Percent used since      CPU usage: CLASP start/last cycle |
| Classes  Percent CLASP start/last cycle  ---------------------------------|
| -------  ------- ----------------------      User       74%         77%  |
|    A       60%       0%          0%          System     24%         23%  |
|    B       40%       0%          0%          Idle        2%          0%  |
|    C      100%       0%          0%                                       |
|                                                                         |
|                                              Class name key             |
|                                              --------------             |
|                                              A = DEFAULT.CLASS          |
| Secondary        Percent used since          B = PRIVILEGED.USERS       |
| Classes   Level  CLASP start/last cycle      C = BATCH.STD              |
| ---------  -----  ----------------------      D = BATCH.LOW              |
|    D         1       0%          0%          E = MARKETING              |
|    F         1       0%          0%          F = MIS                    |
|    E         2       0%          0%          G =                        |
|                                              .                          |
+-------------------------------------------------------------------------+
```

Figure 2-15. Monitor Screen with Class Scheduling Disabled

Since class scheduling is disabled, CLASP reports class percentages as 0%.

(CLASP always reports 0% when settings that enable scheduling or turn on accumulation mode haven't been applied, and when a class is actually getting 0%.) CLASP does report real user, system, and idle figures, though, and these show the system is busy enough for useful monitoring.

For a meaningful display, go to screen 4 "Change processor status", turn accumulation mode on, and apply settings, using the following keystrokes:

```
BREAK/ESC              (Return to "Monitor summary"  screen)
BREAK/ESC              (Return to Main Menu)
4 <nl>                 (Choose "Change processor status" screen)
ACCUMULATE <nl>        (Activate accumulation mode)
Specify logical processor menu number...  1 <nl>   (Do it on DEFAULT.LP)
BREAK/ESC              (Return to Main Menu)
6 <nl>                 (Apply settings, telling system to accumulate)
7 <nl>                 (Return to "Monitor summary" screen)
1 <nl>                 (Return to "Monitor" screen for processor DEFAULT.LP)
```

After a few minutes, the screen resembles Figure 2-16, next.

```
Nov 18, 1985   Monitor processor: DEFAULT.LP        State: A      4:59:33 PM

Primary                Percent used since        CPU usage: CLASP start/last cycle
Classes   Percent      CLASP start/last cycle    ---------------------------------
-------   -------      ----------------------         User      76%          76%
  A         60%          35%          31%              System    23%          22%
  B         40%          28%          30%              Idle       1%           2%
  C        100%          25%          27%

                                                  Class name key
                                                  --------------
                                                  A = DEFAULT.CLASS
Secondary              Percent used since         B = PRIVILEGED.USERS
Classes    Level       CLASP start/last cycle     C = BATCH.STD
---------  -----       ----------------------     D = BATCH.LOW
   D         1            0%           0%         E = MARKETING
   F         1            7%           6%         F = MIS
   E         2            5%           5%         G =
                                                   .
```

Figure 2-16. Monitor Screen with Accumulation Mode On

The display shows that class BATCH.STD is consuming about 25% of the logical processor's time. Since you want to allot it more time than it usually gets, you can use the "Allot action" screen to give more time -- say 31%. At same time, to maintain a total of 100%, adjust other class percentages, subtracting 3% from each of the other classes. So, give DEFAULT.CLASS 32%, PRIVILEGED.USERS 25%, and BATCH.STD 31%.

The same concept applies to classes that you want to get less processing time. The "Monitor processor" screen gives you a frame of reference.

As you examine monitor screens, remember that the state of class scheduling is important. If the state is A (accumulating), this means that scheduling is not in force -- thus the primary figures and reported percentages may vary widely.

Also, if the idle figure (near the bottom) is high, the environment is not appropriate for testing or for classes, as explained under "Why Use It", above. (The point is to monitor your system under typical, heavy load.)

The "CLASP start" figures are based on the time you last executed CLASP -- or moved a job processor to or away from the logical processor.

A deeper, more comprehensive example of monitoring appears as part of Chapter 3, "On-Site Example with Classes and Logical Processors."

Creating and Using CLASP Script and Overview Files

The class and logical processor environment you set up with CLASP is not applied to AOS/VS until

- you set up the environment you want in CLASP and select choice 6 "Apply"; or

- you run CLASP with the /BATCH switch and a script file that sets up the environment. The script file must have been written during a previous CLASP run.

In either case, the environment is temporary; it lasts only while AOS/VS runs. When AOS/VS is shut down, it reverts to the default state, with the mother processor (default 0) connected to logical processor DEFAULT.LP.

So, script files are essential if you want to use classes routinely. They can recreate the class environment easily and consistently when AOS/VS comes up -- perhaps from the UP macro, after any needed JPINITIALIZE commands to the CLI.

Overview files are useful as a general description of the environment; you can print them and post them for users, or file them for future reference.

You must decide on the environment you want before creating a script or overview file. A good way to do it is to experiment with CLASP for awhile -- perhaps using the monitor choice -- then decide on the following items:

- Classes to create

- Logical processors to create

- Allotments of processor time to classes (for the first pass, to monitor, allot 100% and correct later)

- Job processors to move to logical processors

- Logical processor state to select (accumulate to monitor; enable when you want class scheduling to occur)

A list, perhaps similar to the one shown next in Figure 2-17, can help here. Run CLASP and experiment, perhaps using monitoring, and apply changes to the system.

Class and Logical Processor Environment Creation Check List

1. Create class and reassign locality pairs -- Choice 1 (fill in)

Class name: _____ Purpose: _____
Locality pairs: User___Program___, User___Program___, User___Program___

Class name: _____ Purpose: _____
Locality pairs: User___Program___, User___Program___, User___Program___

Class name: _____ Purpose: _____
Locality pairs: User___ Program___, User___Program___, User___Program___

Class name:_____ Purpose: _____
Locality pairs: User____Program___, User___Program___, User___Program___

2. Create logical processor -- Choice 2 (fill in)

LP name: _____ Purpose, classes: _____
LP name: _____ Purpose, classes: _____
LP name: _____ Purpose, classes: _____

3. Allot processor time to classes -- Choice 3 (correct or fill in)

LP name: _____ Class name: _____ Prim/Sec Pct/level: ___
Class name: _____ Prim/Sec Pct/level: ___
Class name: _____ Prim/Sec Pct/level: ___
Class name: _____ Prim/Sec Pct/level: ___
Notes: _____

LP name: _____ Class name: _____ Prim/Sec Pct/level: ___
Class name: _____ Prim/Sec Pct/level: ___
Class name: _____ Prim/Sec Pct/level: ___
Class name:_____ Prim/Sec Pct/level: ___
Notes: _____

LP name: _____ Class name: _____ Prim/Sec Pct/level: ___
Class name: _____ Prim/Sec Pct/level: ___
Class name: _____ Prim/Sec Pct/level: ___
Class name: _____ Prim/Sec Pct/level: ___
Notes: _____

4. Change processor status -- Choice 4 (fill in)

LP name: _____ State (E/D/A): __ Job processor number(s): __ __
Notes: _____

LP name: _____ State (E/D/A): __ Job processor number(s): __ __
Notes: _____

LP name: _____ State (E/D/A): __ Job processor number(s): __ __
Notes: _____

Figure 2-17. Class and Logical Processor Environment Check List

When you've decided on the items shown in Figure 2-17, run CLASP (if it's not running) and select all the nondefault items you want. You're not required to apply changes to the system (choice 6 "Apply"), but doing so will provide an immediate error check.

When you're done, type WRITE <nl>, and specify a script or overview file and pathname. Note that if a file with the name you specify already exists, CLASP will delete and recreate it.

You can use a script file as a starting point for other script files. Implementing classes is a project with many phases, so you'll probably want to modify script files fairly often. For example, take the following phases of development (taken from the class overview near the beginning of the chapter).

- Create classes, assign locality pairs, allot percentages, and turn accumulation mode on. Create script file PHASE1 and leave CLASP.

- Later, with the typical applications environment running, start CLASP with the script file (X CLASP/SCRIPT=PHASE1). In CLASP, apply settings, then use the monitor screen to select better percentages; specify the percentages to CLASP; create script file PHASE2 and leave CLASP.

- Next, with the typical applications environment still running, start CLASP with the latest script file (X CLASP/SCRIPT=PHASE2). In CLASP, enable scheduling and apply settings -- providing the acid test for the class environment.

  If the class environment works well, create script file PHASE3 and use that file routinely to bring up the class environment. If the class environment doesn't work well, experiment until you find one that does -- and create script file PHASE3 and use it routinely as above.

After you're satisfied with the CLASP setting in a script file, you can recreate the class environment contained within the file via the command

XEQ CLASP/BATCH/SCRIPT_FILE=pathname

You'll probably want to test each script file this way after creating it. If it doesn't work (for example, it produces poor results), fix the file before using it routinely in system operations.

Ultimately, you may create script files for several different situations -- each specifying a class environment for a different application, time, and system load.

You can type the WRITE command from any CLASP menu.

An example script writing sequence is shown in Figure 2-18. The sequence for an overview file would be similar.

```
+-------------------------------------------------------------------------+
| Console Dialog                                    Comments on Dialog    |
| ----------------------------------------------    ------------------    |
|                                                                         |
| ) XEQ CLASP <nl>                                  (Start CLASP)         |
| .                                                 (Set up desired       |
| .                                                 class environment)    |
| .                                                                       |
| Enter choice WRITE <nl>                           (Tell CLASP to write) |
| Do you want a script or                                                 |
|             overview file (Script or Overview)?   (What kind of file?)  |
| Enter choice SCRIPT <nl>                          (Want a script file)  |
| Please specify pathname 9.TO.5.SCRIPT.CLASP <nl>  (Specify pathname)    |
|                                                                         |
| Enter choice BYE <nl>                             (Exit via BYE)        |
|                                                                         |
| ) X CLASP/BATCH/SCRIPT FILE=9.TO.5.SCRIPT.CLASP<nl>  (Start CLASP with the|
|                                                   script file; it runs, |
|                                                   then the CLI returns) |
|                                                                         |
| ) X CLASP <nl>                                    (Start CLASP again to |
|                                                   check class use via   |
|                                                   "Monitor" screen)     |
+-------------------------------------------------------------------------+
```

Figure 2-18. Writing and Using a CLASP Script File

Sample Overview File

As mentioned above, overview files can help you and other users keep track of classes and logical processors in your system. A sample overview file follows in Figure 2-19.

```
        Class Assignment and Scheduling Package -- Overview File

 Class Information
 -----------------

                          Program Locality                          Class name key
        0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
     +-----------------------------------------------
   0 |  A  A  C  D  A  A  A  A  A  A  A  A  A  A  A  A   A = DEFAULT.CLASS
   1 |  A  A  C  D  A  A  A  A  A  A  A  A  A  A  A  A   B = PRIVILEGED.USERS
U  2 |  B  B  B  D  B  B  B  B  B  B  B  B  B  B  B  B   C = BATCH.JOBS
s  3 |  B  B  B  D  B  B  B  B  B  B  B  B  B  B  B  B   D = CRITICAL.REPORTS
e  4 |  A  A  C  D  A  A  A  A  A  A  A  A  A  A  A  A   E =
r  5 |  A  A  C  D  A  A  A  A  A  A  A  A  A  A  A  A   F =
   6 |  A  A  C  D  A  A  A  A  A  A  A  A  A  A  A  A   G =
L  7 |  A  A  C  D  A  A  A  A  A  A  A  A  A  A  A  A   H =
o  8 |  A  A  C  D  A  A  A  A  A  A  A  A  A  A  A  A   I =
c  9 |  A  A  C  D  A  A  A  A  A  A  A  A  A  A  A  A   J =
a 10 |  A  A  C  D  A  A  A  A  A  A  A  A  A  A  A  A   K =
l 11 |  A  A  C  D  A  A  A  A  A  A  A  A  A  A  A  A   L =
i 12 |  A  A  C  D  A  A  A  A  A  A  A  A  A  A  A  A   M =
t 13 |  A  A  C  D  A  A  A  A  A  A  A  A  A  A  A  A   N =
y 14 |  A  A  C  D  A  A  A  A  A  A  A  A  A  A  A  A   O =
  15 |  A  A  C  D  A  A  A  A  A  A  A  A  A  A  A  A   P =

================================================================================

 Logical Processor Information
 -----------------------------

Logical Processor Name:  DEFAULT.LP

Primary class definitions:

          A  50%
          B  40%
          C  10%

Secondary class definitions: none

Time interval:  4.0

Attached job processors:        0
Class scheduling state:         Accumulate
................................................................................

Logical processor name:  NIGHT.LP

Primary class definitions:

          A  40%
          D  60%
          . . .
```

Figure 2-19. Sample Overview File

Security Issues

General AOS/VS security issues are explained in How to Generate and Run AOS/VS. However, there are some issues relating specifically to classes. The class security issues fall into the category of denying service: a user may find a way to run in a privileged class, use a disproportionate amount of processor time, and thus deny service to others. The specific class issues are

- User ability to run in a privileged class. A user can do this by changing user locality, by having a privileged default user locality, or by changing program locality.

  Each user's default user locality, and the ability to change locality, are set by PREDITOR in the user's profile. Users need Superuser privilege to run PREDITOR, thus no unprivileged user can change default user locality or permit use of other localities.

  Generally, let each user's default locality start his/her process in an unprivileged class. The default user locality, 0, puts all users in the DEFAULT.CLASS. So, an easy way to keep the defaults unprivileged is to keep each user's default locality as 0 and keep the default class a general-purpose (not privileged) class.

  Be careful with the PREDITOR "Use other localities" question. Unless you know the user must be able to change his/her user locality, the answer should be No (default).

  Progam locality is harder to control than user locality, since access is governed only by ACL, not possession of Superuser privilege. In many systems, people can run SPRED (they have at least execute (E) access to SPRED); and they have read and execute (RE) access to other program files. Without W access to a program file, users can't apply SPRED settings to the file, which means the original program file is safe.

  Even so, RE access to a program may threaten security -- since, with RE access to a program, a user can copy it into his own directory, change the copy's ACL, run SPRED on the copy, change the program locality, then run the copy in a privileged class. A good way to prevent this is to restrict general program access to E, preventing users from copying the program file. Naturally, the ACL of a program's parent directory must not generally allow ACL changes (it must not generally allow W access).

  Or, you can restrict access to SPRED -- by changing the ACL to allow only specific users E access to it.

- User access to CLASP. To create or delete a class or logical processor, the person running CLASP needs Superuser privilege. To apply CLASP settings (interactively or via CLASP's /BATCH switch), the person running CLASP must have System Manager privilege. You can prevent users from changing system settings by restricting System Manager privilege. Generally, only one or two people on the system should have System Manager privilege.

As always, Superuser privilege must be severely restricted if you care about security. Superusers can run PREDITOR and give themselves any privilege.

A user without special privileges can run CLASP in view-only mode by starting CLASP with the /VIEW_ONLY switch. This switch was designed to let multiple processes check class status (perhaps to see the monitor) without changing anything. To prevent users from running CLASP at all, set CLASP's ACL to grant E access only to yourself and other users you want to be able to run it (perhaps in view-only mode).

End of Chapter

Chapter 3

On-Site Example with Classes and Logical Processors

This chapter gives a comprehensive example of class and logical processor use. It will help you understand how the different concepts and programs work together.

The major sections are

- Defining and Running the Applications Environment
- Limits with Standard Scheduling

- Planning the Class Environment (a Step Summary)
- Setting Up the Class Environment (Summary Step 4)

- Using CLASP's Monitor to Refine Class Settings (Summary Steps 5, 6, 7)
- Using CLASP Interactively to Enable Class Scheduling (Summary Step 8)

- Using PED to Check Classes and Logical Processors
- Checking and Automating the Class Environment (Summary Step 9)

Defining and Running the Applications Environment

The sample system and its environment relates to many examples and scenarios shown earlier. The system has the following hardware:

MV/20000 Model 2 computer with two job processors and 16 megabytes of physical memory;

Five gigabytes (billion bytes) of disk storage with a model 6350 disk subsystem (nine DPJ disk units, with three disk controllers);

152 asynchronous console lines (on ten IACs), serving 148 user consoles, two letter-quality printers, and two modem lines;

Two tape units and two data channel line printers;

Network bus adapter (NBA).

The site runs the following software:

- The CEO system, with Information Management and Decision Base. Sixty people, including nontechnical managers and directors, use CEO.

- The Database Management System (DG/DBMS), PRESENT Information Retrieval Facility, and Sort/Merge. The primary applications are inventory control and general ledger. Application programs obtain on-line data from 30 data entry operators in two shifts. Eight

application programmers update and maintain these programs.

Databases are updated interactively, but reports are generated each day in batch. The system is backed up with the DUMP command each night; one full backup is followed by ten incremental backups.

- The XODIAC Network Management System. The computer system uses XODIAC to communicate with others over a local area network (LAN). And CEO uses XODIAC for intersystem mail, and some users access the network directly.

- AOS/VS and CLASP.

The site has run its current applications for six months.

Limits with Standard Scheduling

Generally, management and users are satisfied. However, several people want changes, as follows:

- Six directors want faster response time; and

- The president wants the daily inventory control reports completed earlier. These reports are run in batch during the second and third shifts. Currently, the previous day's report is ready by midafternoon. The president wants it ready by midmorning.

Both needs relate to scheduling. Therefore, using classes and logical processors may help satisfy them. First, though, the company will examine all possible solutions.

Possible Solutions

The company reviews its processing needs to see if major changes (like another computer) are needed. After several meetings, the company concludes that another system would help -- but the cost, in the current fiscal year, puts this last on the list of possible solutions.

Reducing the number of users during the day -- and the complexity of the report at night -- would work. But this would reduce productivity.

After several meetings, the company decides to try changing its current system to satisfy everyone's needs.

The next step is to check for system bottlenecks, in hopes of finding an easy solution. After checking the three hardware components (disk, physical memory, and processor usage), the system manager comes to the following conclusions:

- Disk I/O is not part of the problem (at least 40% of total space remains available on all LDUs, and the DISCO program shows balanced controller load, busy time of less than 50%, and reasonable seek time).

- Physical memory size is probably not part of the problem (PED shows no swaps). But to really verify this, the AOS/VS Performance Package would be needed.

- Job processor (CPU) usage may be part of the problem (CLASP shows 0 to 1% idle time during most of the working day and occasionally at night).

Apparently, memory and disk I/O are ample -- the issue is processor time on the computer.

Tradeoffs for Processor Time

Processor time seems the limiting factor. Some group of users must yield time to management. Luckily, the data entry operators can get by with a little less time (their terminals are managed by IAC multiplexors, which communicate with the computer's job processors and the operators will barely notice a slight reduction in response time).

With processor time available to reassign, it seems that management's goals can be met through process priority: the system manager need only give the directors and report-generating batch jobs higher priority. However, giving the six directors priority might starve all other processes. With standard scheduling, the highest priority ready process always gets control. So, whenever any of the six director's processes needed a processor, it would take over. The directors could monopolize the system, excluding everyone else.

A better solution is to adjust scheduling for the two exceptional cases (directors and report generation) using classes and logical processors. The system manager decides to try this solution.

Planning the Class Environment (A Step Summary)

The next step is to outline -- conceptually -- all the steps needed to set up the environment. This outline follows the one shown near the beginning of Chapter 2. The steps apply to any environment; you can use them as a plan for your own site. They are

1. Define the applications environment without classes.

2. Run the applications environment.

3. Encounter some limit within standard scheduling. Identify it and decide that classes and logical processors can help you.

4. Set up the class environment. This means planning classes and logical processors, and implementing them as follows:

   a. If a class will be based on username, use PREDITOR.

   b. If a class will based on program name, use SPRED.

      For classes based on both username and program name, use PREDITOR

and SPRED.

c. Create classes and logical processors with CLASP. For percentages, use a tentative figure like 100%. Turn on accumulation mode to monitor class use. Create a script file so that you can repeat these steps easily later.

5. Start CLASP with the script file created above; then run your typical applications environment.

6. Use CLASP monitor information to refine class percentages, other class parameters, and/or class-to-logical processor assignment via CLASP, PREDITOR, or SPRED.

   To give a class more time than it's currently getting, increase its allotment (if possible). To give it less time, reduce its allotment. Generally, maintain a sum of 100% for each logical processor.

7. Using the CLASP monitor, compare the percentages used by classes to your goals for those classes.

   If the reported percentages are not close to your goals, return to step 6. If the percentages are close to your goals, continue.

8. Use CLASP interactively to enable class scheduling.

9. See if the environment (interactive and noninteractive processes) is satisfactory.

   If it's not satisfactory, change from enabled to accumulation mode and return to step 6.

   If it is satisfactory, create a CLASP script file and arrange to have CLASP run the script automatically. You're done!

Thus far, in this example, the site has run through steps 1, 2 and 3. The other steps remain, beginning with setting up the class environment (step 4).

Setting up the Class Environment (Summary Step 4)

Setting up the class environment means planning classes and logical processors, editing user profiles (PREDITOR) and program files (SPRED); then creating classes and logical processors, turning accumulation mode on, and creating a script file (CLASP).

Planning Classes

To handle the special needs of the directors and inventory report, the system manager decides on three classes:

- A privileged class, allowed more processing time than is currently allowed, for the six directors;

- A critical reports class, also allowed more processing time than is currently allowed, for the inventory report batch jobs;

- The default class for data entry operators and other users (except the six directors).

Percentage allotments for the classes are unknown; they will be decided later, after CLASP's monitor has given a plausible starting percentage.

Two logical processors will be needed:

1. The default logical processor, for daytime (first shift) operation; it will include the privileged and default classes;

2. A nighttime logical processor, for the second and third shift; it will include the default and critical reports classes.

The logical processor, class, and job processor arrangement will resemble the one shown in Figure 3-1, next.

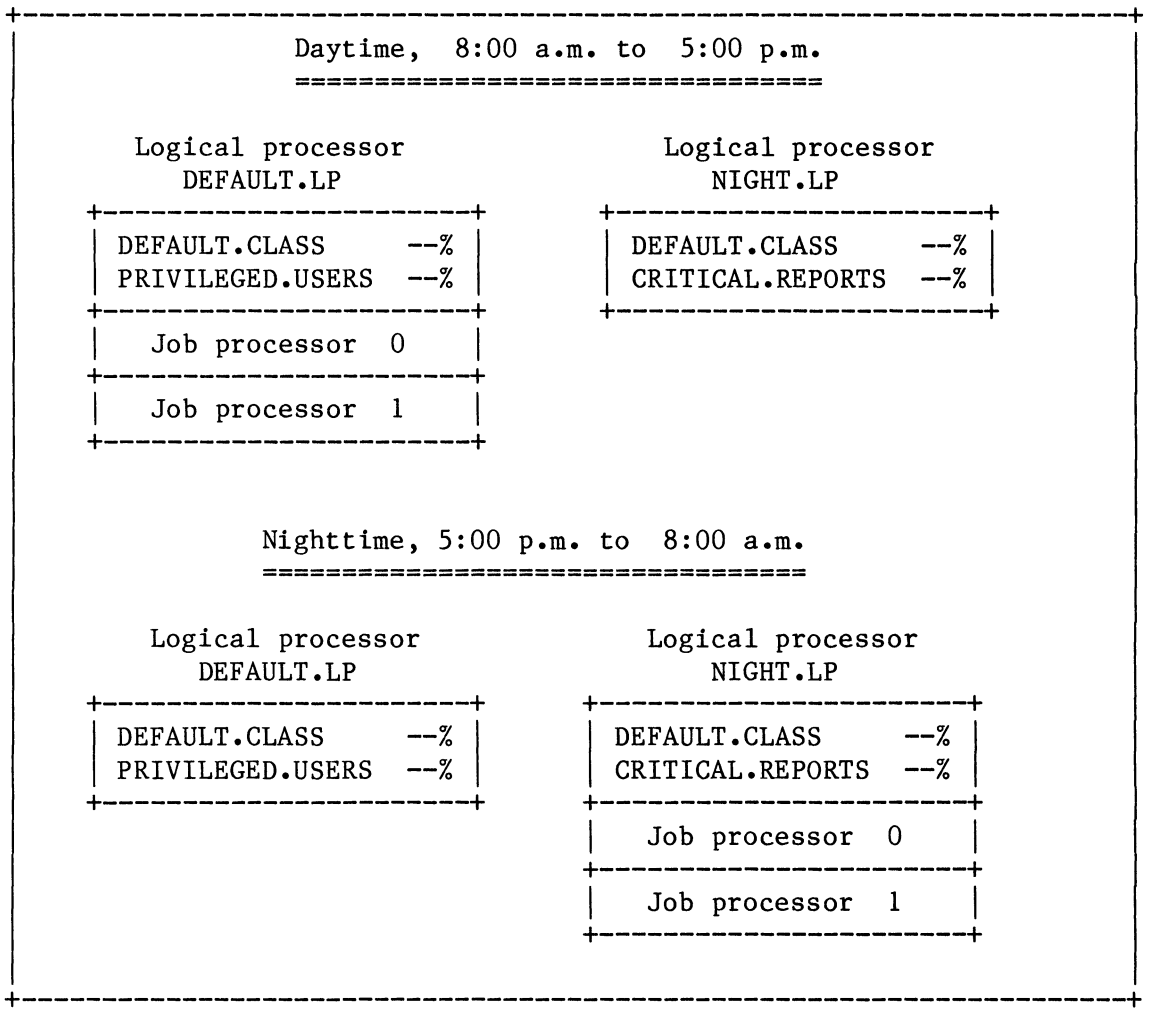

Figure 3-1. Projected Class Allotment for Sample Site

Having planned classes and logical processors, the system manager must implement them. The manager will set up user profiles (PREDITOR) and

program files (SPRED), then create classes and logical processors, turn accumulation mode on, and create a script file (CLASP).

User Profiles -- PREDITOR

One "set" of users is needed: the six directors who want faster response time. They will be privileged users. Since the class is based on username, not program, only PREDITOR, not SPRED, is needed.

The system manager runs PREDITOR and edits each director's user profile. The dialog resembles the following, Figure 3-2.

```
) XEQ PREDITOR <nl>                    (Start PREDITOR)
Command? E <nl>                        (Repeat this sequence of commands
                                        for each director)
Username? SBK <nl>                     (Specify director's username)
.                                      (Take defaults for these questions)
.
.
Default user locality [0]? 0 <nl>      (Start with default user locality)
Use other localities [N]? Y <nl>       (Allow other localities)
User locality (0 - 15) [0]? 1 <nl>     (Allow change to user locality 1)
.                                      (Take defaults for these questions)

                                       (Edit each director's profile)
Command? BYE <nl>                      (Ultimately, leave PREDITOR)
```

Figure 3-2. PREDITOR Dialog at Sample Site

This will bring each director up in the default class, but allow him or her to change locality to 1 -- the privileged class. The directors are CEO users, who don't want to type CLI commands, so their log-on startup macro (LOGON.CLI) will issue the CLI command LOCALITY to change the process from the default class to the privileged class.

Each director's CEO processes will also be privileged (since by default a son process is created with its father's locality). The log-on macro will issue the LOCALITY command conditionally, only from 8:00 a.m. to 5:00 p.m., since, if a director logged on while the nighttime logical processor was running, changing locality to 1 would put him or her in an undefined class. Keeping the user locality 0 between 5:00 p.m. and 8:00 a.m., while the nighttime logical processor is running, will allow the director to run in the default class.

For the privileged plan to work, the correct logical processor must be running. One way to ensure this is with a CLI macro (that can be executed by itself and by the UP macro) that runs CLASP with a script file that sets up the proper environment, based on the time of day. The macro would run CLASP with one script file between 8:00 a.m. and 5:00 p.m, and run CLASP with a different script file at other times.

For the inventory report class, only the program locality -- not the user locality -- pertains. So only SPRED, not PREDITOR, will be needed.

Two programs create the inventory report. Their names are ITEM.PR and FORMAT.PR. The system manager runs SPRED on these programs, answering SPRED questions as follows, in Figure 3-3.

```
)  XEQ  SPRED  ITEM.PR <nl>                          (Start SPRED)
                                                     (It displays menu)
     .
     .
     .
     7. Edit program locality

   Enter choice(s) separated by commas:  7 <nl>   (Do program locality)

   Enter program locality [0]:  1 <nl>            (New locality is 1)
     .
     .
     .
     8. Apply changes to program file

   Enter choice(s) separated by commas:  8 <nl>   (Apply changes to file)
     .
     .
     .

   Enter choice    separated by commas:  BYE <nl>  (Leave SPRED)

   )
   (Dialog is similar for the second program, FORMAT.PR)
```

Figure 3-3. SPRED Dialog at Sample Site

This will force the two programs to run in the critical reports class, which will be a privileged class defined on the nighttime logical processor only (the programs belong to a class that's undefined on the default logical processor).

For this plan to work, the correct logical processor must be running. To ensure this, a CLI macro will run CLASP with a script file to create the proper environment, then it will start the first report-generating program, ITEM.PR, in batch. Using the macro will automatically set up the environment before running the pertinent program. Before running CLASP, the macro will check the time, and will exit without running CLASP or ITEM if the time is between 8:00 a.m. and 5:00 p.m.

Next, the manager will build a base for monitoring using CLASP. This involves creating classes and a logical processor, allotting a tentative amount of processing time to the classes, turning accumulation mode on, and writing a script file. (The step of enabling class scheduling will wait until CLASP's monitor screen has provided feedback on percentage allotments.)

The system manager makes a photocopy of the blank check list in Chapter 2 (Figure 2-17) and notes the desired settings, leaving the percentage figures blank for the time being. A copy of the completed check list follows in Figure 3-4.

```
      Class and Logical Processor Environment Creation Check List

1. Create class and reassign locality pairs -- Choice 1 (fill in)

  Class name: PRIVILEGED.USERS  Purpose: Directors who need faster response.
  Locality pairs: User 1 Program 0-15, User____Program____, User___Program___

  Class name: CRITICAL.REPORTS  Purpose: More CPU for report-creating batch.
  Locality pairs: User 0-15 Program 1, User____Program____, User___Program___

  Class name: DEFAULT.CLASS     Purpose: All processes but directors,reports.
  Locality pairs: User___Program____, User____Program_____, User__Program__

  Class name: _____________     Purpose: ____________________________________
  Locality pairs: User____Program____, User____Program____, User___Program___

2. Create logical processor -- Choice 2 (fill in)

  LP name: NIGHT.LP       Purpose, classes: Run 5pm-8am; report generation.
  LP name: ____________   Purpose, classes: ____________________________
  LP name: ____________   Purpose, classes: ____________________________

3. Allot logical processor time to classes -- Choice 3 (correct or fill in)

  LP name:  DEFAULT.LP  Class name: DEFAULT.CLASS    PrimXXXX  Pct/level: 100
                        Class name: PRIVILEGED.USERS PrimXXXX  Pct/level: 100
                        Class name: _______________  Prim/Sec  Pct/level: ___
                        Class name: _______________  Prim/Sec  Pct/level: ___
  Notes: Percentages are tentative, for revision after monitor._____________

  LP name: NIGHT.LP       Class name: DEFAULT.CLASS   PrimXXXX  Pct/level: 100
                          Class name:CRITICAL.REPORTS PrimXXXX  Pct/level: 100
                          Class name: _____________   Prim/Sec  Pct/level: ___
                          Class name:_____________    Prim/Sec  Pct/level: ___
  Notes: ____________________________________________________________________

  LP name: _____________  Class name: _____________   Prim/Sec  Pct/level: ___
                          Class name: _____________   Prim/Sec  Pct/level: ___
                          Class name: _____________   Prim/Sec  Pct/level: ___
                          Class name: _____________   Prim/Sec  Pct/level: ___
  Notes:_____________________________________________________________________

4. Change processor status -- Choice 4 (fill in)

  LP name: DEFAULT.LP     State (E/D/A): A    Job processor number(s):  0  1
  Notes: LP and script file to be used between 8am and 5pm._______________

  LP name: NIGHT.LP____   State (E/D/A): A    Job processor number(s):  0  1
  Notes: LP and script file to be used between 5pm and 8am._______________

  LP name: ____________   State (E/D/A): __   Job processor number(s): __ __
  Notes: _____________________________________________________________________
```

Figure 3-4. Initial Class and Logical Processor Environment Check List

Creating Classes and a Logical Processor

Having outlined the classes and logical processors needed, the system manager fires up CLASP to create them. The class dialog goes as in Figure 3-5.

Console Dialog	Explanation of Dialog
) X CLASP <nl>	(Start CLASP)
.	(It displays Main Menu)
.	
Enter choice 1 <nl>	(Select "Create" choice)
Create, delete, or modify classes	(CLASP displays "Create"
.	screen; the default
.	class, with name key A,
.	fills the class matrix)
Enter choice (Create, Delete ... CREATE <nl>	(Choose "Create")
Specify new class name: PRIVILEGED.USERS <nl>	(Specify class name)
Do you want to create another class...[Y] <nl>	(Yes, create another)
Specify new class name: CRITICAL.REPORTS <nl>	(Specify next name)
Do you want to create another class...[Y] N <nl>	(No, not another)
Do you want to reassign locality pairs...[Y] <nl>	(Yes, want to reassign)
Specify class name key [C]: B <nl>	(Do privileged class)
Specify user locality: 1 <nl>	(The user locality is 1)
Specify program locality: <nl>	(Use <nl> to specify all program localities, 0 through 15)
Do you want to reassign another locality pair (Y or N)? [Y] <nl>	(Yes, for other class)
Specify class name key [B]: C <nl>	(Do the reports class)
Specify user locality: <nl>	(Use <nl> to specify all user localities, 0 through 15)
Specify program locality: 1 <nl>	(Program locality is 1)
Do you want to reassign another locality pair (Y or N)? [Y] N <nl>	(No, done with pairs)
Enter choice (Delete, Create...)	(Original prompt returns; class matrix appears as follows)

Program Locality

User locality	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	A	C	A	A	A	A	A	A	A	A	A	A	A	A	A	A
1	B	C	B	B	B	B	B	B	B	B	B	B	B	B	B	B
2	A	C	A	A	A	A	A	A	A	A	A	A	A	A	A	A
3	A	C	A	A	A	A	A	A	A	A	A	A	A	A	A	A
4	A	C	A	A	A	A	A	A	A	A	A	A	A	A	A	A
5	A	C	A	A	A	A	A	A	A	A	A	A	A	A	A	A
6	A	C	A	A	A	A	A	A	A	A	A	A	A	A	A	A
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
15	A	C	A	A	A	A	A	A	A	A	A	A	A	A	A	A

Class name key

A = DEFAULT.CLASS
B = PRIVILEGED.USERS
C = CRITICAL.REPORTS
D =
E =
F =
G =
.
P =

Figure 3-5. Creating Classes and Assigning the Locality Pairs

The next steps are to return to the Main Menu via BREAK/ESC, select choice 2 "Create ... processor", and create logical processor NIGHT.LP.

Allotting Processor Time to the Classes

Now, within CLASP, the system manager allots time to the new classes. Not knowing what percentage to specify, the manager allots 100% to each class. The monitor will help refine this. The "Allot" dialog looks like multiple page Figure 3-6, following.

```
Console Dialog                                    Explanation of Dialog
----------------------------------------------    ---------------------

.                                                 (Main Menu display)
.
Enter choice  3 <nl>                              (Select "Allot" choice)
                                                  (CLASP displays "Allot
+-------------------------------------------+     summary" screen)
|   Allot logical processor time to classes ... |
|                                               |
|  1   DEFAULT.LP    INT: 4.0    PRI: A    SEC: |
|  2   NIGHT.LP      INT: 4.0    PRI: A    SEC: |
|                                               |
|  Enter choice  1 <nl>                         |  (Select DEFAULT.LP;
+-------------------------------------------+     CLASP displays "Allot
                                                  action" screen)
+-----------------------------------------------------------------+
|          Allot processor time to classes ...  DEFAULT.LP        |
|                                                                 |
|  Primary                                                        |
|  Class   %     Secondary Class        Class Name Key            |
|  ----------    ---------------        ---------------------     |
|   A    100                            A = DEFAULT.CLASS         |
|                                       B = PRIVILEGED.USERS      |
|                                       C = CRITICAL.REPORTS      |
|                                       .                         |
+-----------------------------------------------------------------+

Enter choice (Primary, Secondary...  PRIMARY <nl>   (Start with primaries)
Action (Modify, Delete ...  MODIFY <nl>             (To allot, modify)
Specify class name key:  A <nl>                     (A is default class)
                      Class percentage:  100 <nl>   (Give class percentage)
Action (Modify, Delete ...  MODIFY <nl>             (Do the other class)
Specify class name key:  B <nl>                     (B is privileged class)
                      Class percentage:  100 <nl>   (Give class percentage)
Action (Modify, Delete ...  BREAK/ESC               (BREAK/ESC exits from
                                                    primary choice; the
                                                    "Allot action" screen
                                                    appears as follows)
+-----------------------------------------------------------------+
|          Allot processor time to classes ...  DEFAULT.LP        |
|  Primary                                                        |
|  Class   %     Secondary Class        Class Name Key            |
|  ----------    ---------------        ---------------------     |
|   A    100                            A = DEFAULT.CLASS         |
|   B    100                            B = PRIVILEGED.USERS      |
|                                       C = CRITICAL.REPORTS      |
|                                       .                         |
+-----------------------------------------------------------------+

Enter choice (Primary, Secondary ...  BREAK/ESC     (Done with DEFAULT.LP;
                                                    leave "Allot action")
```

Figure 3-6. Allotting Classes Time on Logical Processors (continues)

```
+-----------------------------------------------+  (CLASP displays updated
|  Allot logical processor time to classes ...  |  "Allot summary" screen)
|                                               |
|  1  DEFAULT.LP   INT: 4.0    PRI: A B    SEC: |
|  2  NIGHT.LP     INT: 4.0    PRI: A      SEC: |  (NIGHT.LP still has the
|                                               |  defaults)
|  Enter choice  2 <nl>                         |  (Select NIGHT.LP)
+-----------------------------------------------+

                                                    (CLASP displays "Allot
                                                    action" screen)
+-------------------------------------------------------------------+
|          Allot processor time to classes ...  NIGHT.LP            |
|                                                                   |
|  Primary                                                          |
|  Class   %     Secondary Class        Class Name Key              |
|  -------   ---   ---------------      ---------------------------  |
|   A      100                          A = DEFAULT.CLASS           |
|                                       B = PRIVILEGED.USERS        |
|                                       C = CRITICAL.REPORTS        |
|                                       .                           |
+-------------------------------------------------------------------+

Enter choice (Primary, Secondary ...  PRIMARY <nl>   (Start with primaries)
Action (Modify, Delete ...  MODIFY <nl>              (To allot, modify)
Specify class name key:  A <nl>                      (A is default class)
                    Class percentage:  100 <nl>      (Give class percentage)
Action (Modify, Delete ...  MODIFY <nl>              (Do the other class)
Specify class name key:  C <nl>                      (C is reports class)
                    Class percentage:  100 <nl>      (Give class percentage)
Action (Modify, Delete ...  BREAK/ESC                (BREAK/ESC exists from
                                                     primary choice; "Allot
                                                     action" screen appears
                                                     as follows)
+-------------------------------------------------------------------+
|          Allot processor time to classes ... NIGHT.LP             |
|                                                                   |
|  Primary                                                          |
|  Class   %     Secondary Class        Class Name Key              |
|  -------   ---   ---------------      ---------------------------  |
|   A      100                          A = DEFAULT.CLASS           |
|   C      100                          B = PRIVILEGED.USERS        |
|                                       C = CRITICAL.REPORTS        |
|                                       .                           |
+-------------------------------------------------------------------+

Enter choice (Primary, Secondary ...  BREAK/ESC      (You're done with this
                                                     processor; BREAK/ESC
                                                     leaves "Allot action"
                                                     screen)
```

Figure 3-6. Allotting Classes Time on Logical Processors (continued)

```
+-------------------------------------------------------------------------+
|  +-------------------------------------------------+ (CLASP displays updated |
|  |    Allot logical processor time to classes ...  | "Allot summary" screen, |
|  |                                                 | with the two processors) |
|  |  1  DEFAULT.LP   INT: 4.0   PRI: A B    SEC:    |                         |
|  |  2  NIGHT.LP     INT: 4.0   PRI: A C    SEC:    |                         |
|  |                                                 |                         |
|  | Enter choice  BREAK/ESC                         | (Exit to Main Menu)     |
|  +-------------------------------------------------+                         |
+-------------------------------------------------------------------------+
```

Figure 3-6. Allotting Classes Time on Logical Processors (concluded)

This dialog sequence allots logical processor time to the classes.

The next steps are -- in CLASP -- to set up for monitoring by turning accumulation mode on and creating a script file. (CLASP won't really move anything or turn accumulation mode on until settings are applied or CLASP is run with the script file. The script file just provides the basis for monitoring.)

These steps and their dialog is shown as follows, in Figure 3-7.

```
Console Dialog                                    Explanation of Dialog
----------------------------------------------    ---------------------

.                                                 (CLASP Main Menu)
Enter choice  4 <nl>                              (Select "Change
                                                  processor status"; CLASP
+-------------------------------------------+     displays the screen)
|          Change processor status          |
|                                           |
|  Logical Processors   State   Job Processors
|  ------------------   -----   -------------
|                                           |
|  1  DEFAULT.LP         D       0  1       |     (Screen includes job
|  2  NIGHT.LP           D                  |     processor 1 since this
|                                           |     has been initialized)
|  Enter choice (Accumulate, Disable,       |
|             Enable, or Move)  ACCUMULATE <nl>|   (Select "Accumulate")
+-------------------------------------------+

 Specify logical processor menu number...  A <nl>  (Specify all processors)

                                                  (CLASP displays updated
+-------------------------------------------+     "Change processor
|          Change processor status          |     status" screen)
|                                           |
|  Logical Processor    State   Job Processors
|  ----------------     -----   -------------
|                                           |
|  1  DEFAULT.LP         A       0  1       |
|  2  NIGHT.LP           A                  |
|                                           |
|  Enter choice (Accumulate...  WRITE <nl>  |     (Looks good; write a
|                                           |     script file)
|  Do you want a script or overview file ...|
|  Enter choice  SCRIPT <nl>                |     (Choose script)
|  Please specify pathname  8.TO.5.TEST <nl>|     (Type the pathname)
|  Enter choice (Accumulate...  BREAK/ESC   |     (Done with "Change
+-------------------------------------------+     processor status"; exit
                                                  to Main Menu)
 .                                                (Main Menu returns)
Enter choice  BYE <nl>                            (Exit to CLI)
)                                                 (CLI returns)
```

Figure 3-7. Using Change Status to Turn On Accumulation Mode

This sequence creates CLASP script file 8.TO.5.TEST, which will serve as a building block for the next (monitor) CLASP session.

Although this script file is destined for DEFAULT.LP, turning on accumulation mode for NIGHT.LP will do no harm and will eliminate a step when the manager creates the script file for NIGHT.LP. (However, the manager will need this "Change processor status" screen to move both job processors to NIGHT.LP before creating the script file).

Within this CLASP session, classes have been defined, a logical processor created, time allotted on both logical processors, and accumulation mode turned on for both processors.

The next step is the monitor: run CLASP with the script file; then run the typical applications environment.

Using CLASP's Monitor to Refine Class Settings (Summary Steps 5, 6, 7)

The next step is to monitor the classes, with a view toward refining their percentage allotments. No change in scheduling will actually occur, since class scheduling isn't enabled; instead, the logical processor has accumulation mode on.

For monitoring to be meaningful, the system must be running the pertinent applications environment, with the pertinent classes. Thus, while monitoring the DEFAULT.LP, the manager must have all normal daytime applications running. And most (if not all) of the six directors should be using the system as usual, but in the privileged class.

This is true for any logical processor: while it's being monitored, all programs that pertain to the environment must be running in the pertinent classes.

This section shows monitoring of the daytime environment (DEFAULT.LP) only. The NIGHT.LP session involves largely the same steps but will be done at night (say at 8:30 p.m. and 2 a.m., while one of the inventory report programs is running in batch). However, for the NIGHT.LP class environment, job processors 0 and 1 must be moved to NIGHT.LP in CLASP -- mentioned above -- before settings are applied. (Moving job processors isn't needed with DEFAULT.LP, since all initialized job processors are connected to DEFAULT.LP by default.)

To monitor and refine DEFAULT.LP, the system manager arranges for morning applications to run as usual. Both job processors are on line, using standard AOS/VS scheduling (processor 1 has been initialized by the command JPINITIALIZE 1 in the UP macro as customary).

The next step is to ensure that the directors run in the privileged class. Before the directors arrive and log on, the manager edits their log-on macros, inserting a LOCALITY 1 command in each. This command will change each director process's class from default to privileged. The command can do no harm if the class environment isn't running, since -- if the class environment isn't running -- class scheduling won't be enabled. Without class scheduling enabled, processes can't strand themselves, regardless of the locality they change to.

Next, the manager brings up the class and logical processor environment, in which AOS/VS will accumulate statistics, by running CLASP with the script file created yesterday:

```
) XEQ CLASP/BATCH/SCRIPT=8.TO.5.TEST <nl>
)
```

The system is now accumulating information on class use of processor time. When system load reaches normal and the directors have logged on, the manager can run CLASP again -- and monitor the environment in a meaningful way.

CLASP Monitor Dialog

The CLASP dialog to run the monitor follows, in Figure 3-8. CLASP was started with a script file that told the system to accumulate information, and the system is still doing this. So, the manager can simply execute CLASP without a script file.

```
Console Dialog                                    Explanation of Dialog
----------------------------------------------    ---------------------

) X  CLASP <nl>                                   (Run CLASP with script
                                                  file from last session,
.                                                 interactively)
.                                                 (It displays Main Menu)
 Enter choice:  7 <nl>                            (Select "Monitor"
                                                  choice; CLASP displays
                                                  "Monitor summary"
+-----------------------------------------------+ screen)
|    Monitor logical processors -- summary      |
|                                               |
|    Logical Processors   State                 |
|    ------------------   -----                 |
|                                               |
|    1   DEFAULT.LP       A                     |
|    2   NIGHT.LP         A                     |
|                                               |
| Enter choice (number of entry)  1 <nl>        | (Select DEFAULT.LP;
+-----------------------------------------------+ CLASP displays "Monitor
                                                  processor" screen:)
+-------------------------------------------------------------------+
|...         Monitor logical processor: DEFAULT.LP   State: A   ...  |
|-------------------------------------------------------------------|
|                                                                   |
|Primary            Percent used since      CPU...CLASP start/last cycle|
|Classes   Percent  CLASP start/last cycle  ---------------------------|
|-------   -------  ----------------------   User        76%         76%|
|   A       100%        76%         72%     System      23%         22%|
|   B       100%        24%         28%     Idle         1%          2%|
|                                                                   |
|                                        Class name key             |
|                                        --------------             |
|                                        A = DEFAULT.CLASS          |
|Secondary          Percent used since   B = PRIVILEGED.USERS       |
|Classes    Level   CLASP start/last cycle C = CRITICAL.REPORTS     |
|---------  -----   ----------------------  .                       |
|   .         .          .          .      .                       |
+-------------------------------------------------------------------+

 BREAK/ESC                                        (Got the data needed,
                                                  76% and 24%. Exit from
                                                  DEFAULT.LP "Monitor
+-----------------------------------------------+ processor" screen)
|       Monitor logical processor -- summary    |
|       .                                       |
|  Enter choice ...  BREAK/ESC                  | (Exit from "Monitor
+-----------------------------------------------+ summary" screen to
.                                                 Main Menu)
```

Figure 3-8. Monitoring and Adjusting Class Allotments (continues)

```
.                                                    (It displays Main Menu)
 +-------------------------------------------------+
 |                  Main Menu                      |
 |     .                                           |
 |     3  Allot logical processor time to classes  |
 |     .                                           |
 |  Enter choice  3 <nl>                           |  (Select "Allot" choice;
 +-------------------------------------------------+  CLASP displays "Allot
 +-------------------------------------------------+  summary" screen)
 |     Allot ... time to classes -- summary        |
 |      .                                          |
 |  Enter choice (number of entry)  1 <nl>         |  (Choose "Allot action"
 +-------------------------------------------------+  screen for DEFAULT.LP)
 +-------------------------------------------------+
 |    Allot ... time to classes ... DEFAULT.LP     |
 |    .                                            |  (Select primary; modify;
 |    .                                            |  add 10% to privileged
 |    .                                            |  class (make it 34%);
 |                                                 |  subtract 10% from
 |                                                 |  default class (make it
 |                                                 |  66%).
 |  Enter choice (Primary ...)  BREAK/ESC          |  (Exit from "Allot
 +-------------------------------------------------+  action" screen)

 Enter choice (number of entry)  BREAK/ESC          (Exit from "Allot
                                                     summary" screen to Main
                                                     Menu)
 +-------------------------------------------------+
 |                  Main Menu                      |
 |    .                                            |
 |    6  Apply CLASP settings to the ... system    |
 |    .                                            |
 |  Enter choice  6 <nl>                           |  (To test settings
 +-------------------------------------------------+   interactively, apply
                                                      them...)
 Enter choice  WRITE <nl>                           (The settings work;
                                                     create a script file)
 Create script or overlay file ...  SCRIPT <nl>     (Choose script file)
 Specify pathname...  8.TO.5.ACCUMULATE.ON <nl>     (Type the filename)
 Enter choice  BYE <nl>                             (Leave CLASP for CLI)
 )  X  CLASP/BATCH/SCRIPT=8.TO.5.ACCUMULATE.ON <nl>  (As a second test, run
                                                     the script
                                                     noninteractively)
 )                                                   (It runs)
```

Figure 3-8. Monitoring and Adjusting Class Allotments (concluded)

In this step (which includes steps 5, 6, and 7 in the step overview), the system manager used CLASP's monitor to decide on reasonable class allotments -- and created a script with those allotments. However, the script doesn't actually enable class scheduling; it turns on accumulation mode.

The next step is the acid test -- enabling class scheduling.

Using CLASP Interactively to Enable Class Scheduling (Summary Step 8)

Thus far, class scheduling hasn't been enabled. In this step, the manager enables it. The dialog to enable and apply follows in multiple-page Figure 3-9.

```
Console Dialog                                          Explanation of Dialog
--------------                                          ---------------------

 ) X  CLASP/SCRIPT=8.TO.5.ACCUMULATE.ON <nl>            (Run interactively
                                                        with the script file
                                                        that turns accumulation
                                                        mode on)
 .                                                      (CLASP Main Menu)
 Enter choice  4 <nl>                                   (Select "Change
                                                        processor status";
                                                        CLASP displays screen)
 +-------------------------------------------+
 |          Change processor status          |
 |                                           |
 |  Logical Processors   State   Job Processors
 |  ------------------   -----   ------------
 |                                           |
 |  1  DEFAULT.LP         A       0  1       |  (Again, CLASP shows job
 |  2  NIGHT.LP           A                  |  processor 1 since this
 |                                           |  has been initialized)
 |  Enter choice (Accumulate, Disable,       |
 |             Enable, or Move)  ENABLE <nl> |  (Want to enable)
 +-------------------------------------------+

 Specify logical processor menu number...  1 <nl>       (Specify DEFAULT.LP)

                                                        (CLASP displays updated
 +-------------------------------------------+          "Change processor
 |          Change processor status          |          status" screen)
 |                                           |
 |  Logical Processors   State   Job Processors
 |  ------------------   -----   ------------
 |                                           |
 |  1  DEFAULT.LP         E       0  1       |
 |  2  NIGHT.LP           A                  |
 |                                           |
 | Enter choice (Accumulate...  BREAK/ESC    |  (Exit from "Change
 +-------------------------------------------+  processor status"
                                                  screen to Main Menu; it
                                                  displays Main Menu)
 +-------------------------------------------+
 |                Main Menu                  |
 |                                           |
 |   .                                       |
 |   6  Apply CLASP settings to the ... system
 |   .                                       |
 |  Enter choice  6 <nl>                     |  (The moment of truth;
 +-------------------------------------------+  apply the settings)
```

Figure 3-9. Enabling Class Scheduling Interactively (continues)

```
                                                    (Check that processes
                                                    are still running and
                                                    that users are happy)
  Enter choice  WRITE <nl>                          (Create new script file)
  Create script or overlay file ...  SCRIPT <nl>
  Specify pathname...  8.TO.5.CLASP.SCRIPT <nl>     (This is the workhorse
                                                    daytime script file)
  Enter choice  BYE <nl>                            (Leave CLASP)
  )  X  CLASP/BATCH/SCRIPT=8.TO.5.CLASP.SCRIPT <nl>  (Try the script file
                                                    noninteractively)
  )                                                 (It works; DEFAULT.LP
                                                    is complete!)
```

Figure 3-9. Enabling Class Scheduling Interactively (concluded)

Class scheduling has been enabled on logical processor DEFAULT.LP (and settings applied to the system); the class environment works; and a script file to recreate the environment has been created. This completes creation and initial testing on DEFAULT.LP. The manager tries another monitor run to make sure percentages are reasonable. Then, the daytime class environment is complete.

Next, the system manager will create the nighttime environment. The procedure is similar to the one shown in Figure 3-8 earlier, starting with the script file 8.TO.5.ACCUMULATE.ON.

Before monitoring, the system manager must use CLASP Main Menu choice 4 "Change processor status" to move job processors 0 and 1 to logical processor NIGHT.LP, then create a NIGHT.LP script file. Then the manager will run the script file interactively, check the monitor, and adjust percentages. Next, the manager will test NIGHT.LP by enabling class scheduling and applying settings, and -- if all goes well -- create a NIGHT.LP script file.

Using PED to Check Classes and Logical Processors

The PED utility can display class-related information: a process's user locality, program locality, the class the process is running in, and any processes that can run in the mother processor only (PMGR and any ?IDEF-issuing process always run in the mother only). The switches for this are /ULOCALITY (user locality), /PLOCALITY (program locality), /CLASSNAME (class name), and /MPROCESSOR (mother-only processor).

A macro to show user locality, program locality, class name, mother-only processes, and most other default information, would contain the following lines:

```
PED/CYCLE=10/PID/USER/PROGRAM/ELAPSED/CPU/BS/ULOCALITY/PLOCALITY&<nl>
/CLASSNAME/IO/FTA/MPROCESSOR
```

(This omits only the /PROCESS, /SH7, /US7, and /WSS switches from the default PED display.)

A typical PED display, in the sample site while the DEFAULT.LP above is running, might look like Figure 3-10, next. Notice that PED displays only the first 10 characters of the class name.

PID	USERNAME	PROGRAM	ELAPS	CPU	MP	UL	PL	CLASSNAME	I/O	FTA
1	PMGR	PMGR	2-06	78:02	Y	0	0	DEFAULT.CL	58	9389
3	OP	EXEC	2-06	22.55		0	0	DEFAULT.CL	8686	1327
4	OP	XLPT	2-06	20.40		0	0	DEFAULT.CL	761	46
9	OP	PED	5.00	0.20		0	0	DEFAULT.CL	0	57
12	SALLY	CEO_CP	4/17	13.34		1	0	PRIVILEGED	690	6441
13	MARC	CEO_CP	1/23	3.02		0	0	DEFAULT.CL	155	1441
14	JACK	F77	20.00	0.37		0	0	DEFAULT.CL	9	122
45	SBK	CEO_CP	2/45	7.81		1	0	PRIVILEGED	504	5667
.	.	.	.	.		.	.	.	.	.

Figure 3-10. PED Display with Classes, Sample Site

The PED display shows most processes in the default class -- and two directors running CEO in the privileged class.

Checking and Automating the Class Environment (Summary Step 9)

With the class environment working, the system manager can automate its operation. For the daytime environment, the steps involve

- Editing the UP.CLI macro to run the daytime CLASP script file automatically during the daytime (8:00 a.m. to 5:00 p.m.).

- Editing each privileged director's log-on macro to change locality to 1 (privileged class) conditionally between 8:00 a.m. and 5:00 p.m. (If the macros didn't check the time, and a director logged on in an environment where the privileged class was undefined, the system would reject the LOCALITY 1 command, producing a disconcerting error message -- and the director would run in the default class.

For the nighttime class environment, the steps involve

- Editing the UP macro again, to bring up NIGHT.LP.

- Changing the macros that run programs ITEM and FORMAT in batch so they run CLASP with the correct nighttime script file, then run programs ITEM and FORMAT, using a PROCESS/LOCALITY=1 command to start each program.

As time passes, the manager notes the effect of the class environment -- not only the reaction of the directors and completion time of the inventory reports, but the reaction of the data entry operators, application programmers, and all users.

As problems or other exceptional needs arise, the manager can adjust the class environment -- and use other management tools -- to handle them.

End of Chapter

Chapter 4

Class-Related Errors and Error Conditions

This chapter explains error conditions and situations that can occur when you work with classes. The major sections are

- Class-Related Error Messages and Recovery
- Class-Related Error Conditions without Error Messages

Class-Related Error Messages and Recovery

Two types of error messages are reported. They are

Errors that occur while you're running CLASP (reported by CLASP, via CLASP error messages); and

Errors that occur when you run CLASP with a script file and the /BATCH switch (reported by AOS/VS, via AOS/VS error messages).

. CLASP and AOS/VS sometimes use different text to report the same error. Generally, CLASP's messages are more useful because they occur immediately after you make errors. Also, they're easier to read because they're in upper and lowercase.

AOS/VS error messages occur when you run CLASP with a script file and the /BATCH switch. They can't pinpoint the source of the error; you must backtrack through the original CLASP session to find the error. AOS/VS messages are in uppercase only. As you develop each class environment, you'll see CLASP's messages until you test the script file using the /BATCH switch from the CLI. Then, from the CLI, you'll see AOS/VS's messages.

Generally, you can check for errors in a script file by running CLASP with the script file name but without the /BATCH switch. Then, check screens whose setting(s) may have caused the error. If the problem isn't obvious, try creating an overview file; then print it and examine it for inconsistencies. After you discover the problem, correct it in CLASP. Then create a script file and specify the same name as the original, erroneous script file -- to overwrite the erroneous version.

Other class-related errors can occur when a user logs on (if class scheduling is enabled and the user comes up in a class that has no time allotted), or when a user issues a PROCESS or LOCALITY command that puts one of his/her processes in a class that has no time allotted. Handling this error is described under error message "PROCESS'S CLASS NOT SCHEDULABLE ON AN ACTIVE LP" in the following section.

The following table lists class-oriented error messages alphabetically, then explains the source, possible causes, and recovery action.

Table 4-1. Class-Related Error Messages (continues)

Message	Source, Possible Cause(s), and Action
Applying these settings would strand CLASP -- correct and retry.	From CLASP, when you apply settings or after you execute CLASP with the /BATCH and /SCRIPT_FILE switches. The settings specified within CLASP (or in the script file) would strand the CLASP process. CLASP has not applied the settings. Review the steps in CLASP that led to this situation and identify the error. Correct the error condition; or if the error originated in a script file, fix the script file. This error condition won't occur if you let the default class keep some time on every logical process -- and you let the CLI process from which you run CLASP keep default localities (user and program localities of 0).
ATTEMPT TO EXCEED MAXIMUM LP COUNT (System error code is ERMLP)	From AOS/VS. The script file told CLASP to create a 17th logical processor. The maximum number allowed (including DEFAULT.LP) is 16. The error was probably in CLASP screen 2 "Create ...processor". Fix the script file using that screen.
ATTEMPT TO RELEASE LAST JP ATTACHED TO AN LP (System error code is ERLJP)	From AOS/VS. From the CLI, someone tried to release (JPRELEASE) the last (perhaps only) job processor attached to a logical processor. For this to happen without an error message, the JPRELEASE switch /LAST is required.
CALLER NOT PRIVILEGED FOR THIS ACTION, xxx	From AOS/VS. You, or a user, tried to activate a privilege (xxx) not allowed in the user profile. If this message occurs after you (or your process) issue a JPINITIALIZE command, it means you lack the System Manager privilege, needed to initialize job processors (and create classes and logical processors).

Table 4-1. Class-Related Error Messages (continued)

Message	Source, Possible Cause(s), and Action
CALLER NOT PRIVILEGED FOR THIS ACTION (continued)	If this message occurs when you run CLASP, it means you lack System manager and/or Superuser privilege. To create classes, you need System Manager privilege. To update CLASP name files in :PER, you need Superuser privilege. If this message occurs after you (or a user) type a LOCALITY or PROCESS/LOCALITY command, it means you or the user lacks the "Use other localities" privilege. In any of these cases, for the failed action to succeed, you must run PREDITOR on the pertinent profile and give the needed privileges. Then log off, log on again (or have the user do so), and retry the operation. In situations other than the ones above, this message indicates the lack of other privileges, described under this message in the error chapter of How to Generate and Run AOS/VS.
CANNOT DELETE LP WITH JP ATTACHED (System error code ERJPA)	From AOS/VS. The script file told CLASP to delete a logical processor that has a job processor attached to it. Before the logical processor can be deleted, the job processor must be moved to another logical processor (via CLASP menu choice choice 4 "Change processor status"). You might want to review the whole script file (run CLASP with the script filename without the /BATCH switch, create an overview file; then fix the script file via CLASP.)
CANNOT DELETE CLASS 0 (System error code ERCL0)	From AOS/VS. The script file told CLASP to delete class 0 (DEFAULT.CLASS). This class is part of AOS/VS and cannot be deleted. Fix the script file via CLASP menu choice 1 "Create ... classes".

Table 4-1. Class-Related Error Messages (continued)

Message	Source, Possible Cause(s), and Action
CANNOT DELETE LP 0 (System error code ERLP0)	From AOS/VS. The script file told CLASP to delete logical processor 0, DEFAULT.LP. This processor is part of AOS/VS and cannot be deleted. Fix the script file via CLASP menu choice 2 "Create ... processor".
CLASS DOES NOT EXIST (System error code ERCNE)	From AOS/VS. The script file told CLASP to access a class that doesn't exist. The class was not created via CLASP's "Create ... classes" screen. Fix the script file via CLASP choice 1 "Create ... classes".
Class does not exist -- please try again.	From CLASP. The class doesn't exist. Create it via choice 1 "Create..classes".
CLASS IN USE (System error code ERCLU)	From AOS/VS. The script file told CLASP to delete a class that's defined in the class matrix. This is illegal. Fix the script file via steps in the message "Deleting this class is not allowed..."
Deleting the default class is not allowed -- please try again	From CLASP. You tried to delete DEFAULT.CLASS. This class is built into AOS/VS and cannot be deleted. If you need another class, use choice 1, "Create ... classes" to create it.
Deleting the default logical processor is not allowed -- please try again	From CLASP. You tried to delete DEFAULT.LP. This logical processor is part of AOS/VS and cannot be deleted. If you need another logical processor, create one via 2 "Create ... processor." If 16 processors already exist, delete one to make room for the new one.
Deleting this class is not allowed because class is defined in the matrix. Press NEW LINE or specify a name key for substitution	From CLASP. You tried to delete a class that's defined in the matrix. This is not allowed because it might strand active classes. You can either create and use a class (press NEW LINE) or substitute (copy) the target class's locality pairs to another class (specify the key for substitution).

Table 4-1. Class-Related Error Messages (continued)

Message	Source, Possible Cause(s), and Action
Deleting this LP is not allowed because a job processor (JP) is connected to it. You must first move the JP to a different LP -- please try again.	From CLASP. You tried to delete a logical processor that's active (has a job processor connected to it). If you're sure you want to delete the logical processor (check its classes with the "Allot action" screen), move its job processor to another logical processor (choice 4 "Change processor status"), then try to delete it again.
Illegal character in name -- please try again	From CLASP. You typed an illegal class or logical processor name. Legal names have 1 to 16 filename characters. Retry.
ILLEGAL HIERARCHY LEVEL / PERCENTAGE PAIR (System error code ERHLP)	From AOS/VS. The script file told CLASP put a class in an illegal secondary level (levels are 0 through 15); or it told CLASP to allot an illegal percentage (percentage must be a number and should range from 0 through 100). Fix the script file using CLASP's "Allot action" screen.
ILLEGAL LOCALITY VALUE (System error code ERILV)	From AOS/VS. The script file told CLASP to assign an illegal locality value (must be between 0 and 15). Fix the script file via CLASP menu choice 1 "Create ... classes".
ILLEGAL OPTION REQUESTED	From AOS/VS, this reports an error in a JP-series hardware instruction. The error can occur after the CLI command JPINITIALIZE or JPRELEASE. This error indicates wrong use of an instruction; it shouldn't occur. For action, see "JP NOT STOPPED" message.
INVALID CLASS ID (System error code ERICI)	From AOS/VS. The script file told CLASP to create a class, but the name given was illegal. (This error should have been caught by CLASP when the class was created; you might want to submit a Software Trouble Report.)
Invalid class name key -- please try again	From CLASP. You specified an illegal class name key (legal keys are letters A though P). Retry.

Table 4-1. Class-Related Error Messages (continued)

Message	Source, Possible Cause(s), and Action
INVALID CLASS PERCENTAGE (System error code ERCPC)	From AOS/VS. A script file told CLASP to assign an illegal percentage. Run CLASP with this script file and omit the /BATCH switch; check the "Allot action" screen for each active logical processor; then correct the percentage allotment.
INVALID CPU MODEL NUMBER (System error code ERUCP)	From AOS/VS. The CPU (job processor) has an invalid model number. Reload microcode from MV/System media as described in How to Load and Generate AOS/VS.
Invalid entry -- please try again	From CLASP. Your answer had the wrong form; perhaps you typed a number when CLASP wanted a name. Read CLASP's prompt carefully, then retry.
INVALID HIERARCHICAL LEVEL (System error code ERIHL)	From AOS/VS. The script file told CLASP to put a class in the secondary category but specified an illegal level. The level must be a number, and only numbers from 1 through the next unused level are legal. Fix the script file using the "Allot action" screen.
Invalid interval (must be between .1 and 10 seconds) -- please try again	From CLASP. You specified an illegal time interval for this logical processor. Retry.
INVALID JPID (System error code ERIJP)	From AOS/VS. The job processor ID given in a JPINITIALIZE command was wrong, or the job processor doesn't exist. Reissue the JPINITIALIZE command with the correct number as an argument; for example, JPINITIALIZE 1 Correct the command in the macro, if this applies.
Invalid { program } locality { user } (must be between 0 and 15) -- please try again.	From CLASP. You specified an illegal program or user locality. Retry.

Table 4-1. Class-Related Error Messages (continued)

Message	Source, Possible Cause(s), and Action
INVALID TIME INTERVAL (System error code ERITI)	From AOS/VS. The script file told CLASP to assign an invalid interval. Fix the script file interval setting using CLASP's "Allot action" screen.
Invalid user locality ...	From CLASP. See "Invalid program locality" message.
JP ALREADY ATTACHED TO LP (System error code ERJAA)	From AOS/VS. The script file told CLASP to move a JP to a logical processor, but the JP was already connected to the logical processor. You can fix the script file using CLASP menu choice 4, "Change processor status".
JP ALREADY INITIALIZED (System error code ERJAI)	From AOS/VS. Someone (or perhaps the UP macro) issued a JPINITIALIZE command to a job processor that's already initialized. If you typed the wrong name, try again, and/or correct the command in the macro. If the job processor name was correct, the processor is already initialized; skip the JPINITIALIZE command.
JP FAILED	From AOS/VS, this reports an error in a JP-series hardware instruction. The error can occur after the CLI command JPINITIALIZE. The job processor is faulty and can't run. You may want to call your DG support organization.
JP IS IN A BAD STATE (System error code ERJPS)	From AOS/VS, from JPINITIALIZE command. The job processor has not been initialized by AOS/VS, yet it's running. Perhaps it's running a different operating system, like a diagnostics system. AOS/VS can't initialize the job processor until it's halted. You may want to use the SCP to halt and/or reset the job processor; then retry the JPINITIALIZE command. Or, if the job processor is running diagnostics, wait until the diagnostics finish. (Generally, it's risky to run AOS/VS in one job processor and a different operating system in another.) If you determine that nothing should be running in the job processor, call your DG support organization.

Table 4-1. Class-Related Error Messages (continued)

Message	Source, Possible Cause(s), and Action
JP NOT INITIALIZED (System error code ERJNI)	From AOS/VS, after you use CLASP to apply settings (either interactively or via a script file). You can't move the job processor because it's not initialized. Use the CLI command JPINITIALIZE 1 to initialize the job processor. Then rerun CLASP and apply settings the same way you did before. This message can also occur if you try to release (JPRELEASE) a job processor that's not initialized. This error doesn't relate to CLASP operations; but if it occurs, ignore it and go on with your next operation.
JP NOT RUNNING	From AOS/VS, this reports an error in a JP-series hardware instruction. The error can occur after the CLI command JPINITIALIZE. The job processor is not running, although AOS/VS tried to start it. The job processor may be faulty; you may want to run diagnostics and/or contact your DG support organization.
JP NOT STOPPED	From AOS/VS, this reports an error in a JP-series hardware instruction. The error can occur after the CLI command JPRELEASE. The job processor is still running after AOS/VS tried to release it. Normal shutdown may not be possible; if not possible, you must force shutdown. For action, see the preceding message.
JP RUNNING ONE OR MORE SYSTEM TASKS (System error code ERJST)	From AOS/VS. You tried to release (JPRELEASE) the mother job processor, without which AOS/VS can't run. If you made a typing error, retry. Otherwise, if you must release this processor, you must shut down AOS/VS.
LCS ERROR	From AOS/VS, this reports an error in a JP-series hardware instruction. The error can occur after the CLI command JPINITIALIZE, when AOS/VS tries to load microcode (it uses the LCS instruction). The job processor may be faulty. For action, see "JP NOT RUNNING" message.

Table 4-1. Class-Related Error Messages (continued)

Message	Source, Possible Cause(s), and Action
Level number not available -- please try again	From CLASP. When you place a class in the secondary category, only levels with numbers 1 though the highest unused number are valid. Retry.
LP ALREADY EXISTS (System error code ERLAI)	From AOS/VS. The script file told CLASP to create a logical processor that already existed. You can fix the erroneous setting via the CLASP menu choice 2 "Create ... processor".
LP DOES NOT EXIST (System error code ERLNE)	From AOS/VS. The script file told CLASP to access a logical processor that doesn't exist. The failed access could involve class allotment ("Allot summary" screen), moving a procesor or enabling scheduling (4 "Change processor status" screen), or the monitor (7 "Monitor summary" screen). Run CLASP with the offending script file but without the /BATCH switch and create an overview file. The overview file may show what went wrong. Then fix the CLASP settings and create a new script file.
Maximum number of { classes / logical processors } already exists -- please try again	From CLASP. The maximum number of classes of logical processes (including the default of each) is 16. This means you can create up to 15 classes and logical processors. Try to do what you need by modifying an existing class or processor. If you can't do that, delete a little-used one, then try the CREATE command again.
MICROCODE IS INCOMPATIBLE WITH CURRENT SYSTEM (System error code ERCMM)	From AOS/VS. The microcode file specified for a job processor is invalid. Similar to message "MICROCODE FILE MUST BE REV n OR GREATER" in How to Generate and Run AOS/VS. You must specify a different microcode filename and/or change the default filename; if this fails, you may need to reload microcode from your MV/n system media.

Table 4-1. Class-Related Error Messages (continued)

Message	Source, Possible Cause(s), and Action
NO JP STATE BLOCK DEFINED	From AOS/VS, this reports an error in a JP-series hardware instruction. The error can occur after the CLI command JPINITIALIZE or JPRELEASE. This error indicates wrong definition of a job processor; it shouldn't occur. For action, see "JP NOT STOPPED" message.
NON-EXISTENT JP	From AOS/VS, this reports an error in a JP-series hardware instruction. The error can occur after the CLI command JPINITIALIZE or JPRELEASE. According to the hardware, the job processor you specified doesn't exist. This message may mean you made a typing mistake, typing the wrong number. Retry. If this isn't the problem, see "JP NOT STOPPED" message.
NOT A MULTI-PROCESSOR SYSTEM (System error code ERNMP)	From AOS/VS. You issued a JPINITIALIZE command to initialize another job processor but your computer system doesn't have a second job processor. Do nothing; the JPINITIALIZE command is useful only on a computer with multiple job processors.
PRIVILEGE HELD EXCLUSIVELY BY ANOTHER PROCESS (System error code ERPVX)	From AOS/VS. You tried to run CLASP without the /VIEW_ONLY switch, but some other process (maybe another CLASP process) has System Manager privilege turned on exclusively. Use the ? macro to check for other processes that might have System Manager turned on exclusively. CLASP, when run without the /VIEW_ONLY switch, tries to turn it on exclusively. If the other process using the privilege must keep running, you can run CLASP with the /VIEW_ONLY switch.

Table 4-1. Class-Related Error Messages (concluded)

Message	Source, Possible Cause(s), and Action
PROCESS'S CLASS NOT SCHEDULABLE ON AN ACTIVE LP (System error code ERAVP)	From AOS/VS. Class scheduling is enabled, and the process that you (or a user) tried to create does not belong to a class that has processor time scheduled. The system is preventing creation of a stranded process. The cause may be a PROCESS /LOCALITY switch that would put the new process in a class for which there is no time. Whatever the cause, the process can't run as specified in the current class environment. To let it run, you must change either the PROCESS command or class environment, perhaps by running CLASP with a different script or disabling class scheduling. (The system checks for processor time allotment whenever a process creates another. If a new process can't be scheduled, the system won't create it. It's possible for this to happen when a user logs on (when EXEC tries to create the user process). To fix it, arrange a class environment in which the user can log on -- perhaps by running a different CLASP script file or disabling class scheduling.)
That action is not allowed; you specified view only	You can't make changes via CLASP when it was executed with the /VIEW_ONLY switch. If you need to change settings, leave CLASP and rerun it without the VIEW_ONLY switch. System Manager and Superuser privilege are needed to change settings.

Class-Related Error Conditions without Error Messages

Several error situations can arise without any class-related error message. They are

1. Situations where a CLASP operation strands a process.

2. Situations in which interactive user processes belong to a class that exhausts its pe e ta

Situation 1 -- A CLASP Operation Strands a Process

There are several ways you can inadvertantly strand a process with CLASP. The most common are

- Eliminating a class's processor time via the "Allot action" screen. For example, you reduce class XX's percentage to 0 and -- after settings are applied -- every process belonging to this class freezes. From within CLASP, this would be an obvious error. Via a script file, however, it isn't so obvious an error.

- Moving a job processor from the only logical processor that has time scheduled for a class (via the choice 4 "Change processor status"). How this can happen is explained in Chapter 2. However, we repeat the example here:

  - AOS/VS comes up as usual, with job processor 0 connected to DEFAULT.LP.

  - Via CLASP, you create class XXX, user locality 15 and program locality 0. Then, you create logical processor MYPROC. You allot XXX time on DEFAULT.LP but don't allot it time on MYPROC.

  - A user process comes up -- no matter how -- in user locality 15 and program locality 0. This means it runs in class XXX.

  - Via CLASP -- perhaps via a script file -- class scheduling on both processors is enabled. And, job processor 0 is moved to logical processor MYPROC. This deprives DEFAULT.LP of its job processor.

    Since class XXX isn't allotted time on any other processor, its processes can't be scheduled. The user process running in class XXX becomes stranded. The process acts as if it were frozen. It can't run again until class scheduling is disabled on MYPROC, or a job processor is moved to an LP where class XXX can run (for example, if you moved job processor 0 back to DEFAULT.LP).

A stranded process is treated the same way as a blocked process. AOS/VS maintains the process's state as long as AOS/VS runs. PED displays the process's status as active (not blocked -- B -- or swapped

-- S). At system shutdown, the process will be terminated along with all others.

Situation 2 -- A Class Exhausts Its Percentage Allotment

When a class exhausts its percentage allotment, all processes in the class appear frozen until the next interval. This periodic delay frustrates and annoys interactive users.

You can improve things by reducing the length of the time interval. The default interval is 4 seconds -- but for a logical processor with a class of interactive processes that often exhaust its percentage, you should choose an interval like 1.5 or 1 second. Use CLASP's menu choice 3 "Allot", the "Allot action" screen, to change the interval.

End of Chapter

Glossary

Here are definitions for terms used in the AOS/VS class and logical processor environment.

child processor A job processor (physical processor) other than the default (mother) processor. Each child processor must be initialized (JPINITIALIZE command) before AOS/VS can use it. The concept of mother and child processors applies only in computers with more than one job processor. See also mother processor.

CLASP Class Assignment and Scheduling Package -- topic of this book.

class A set of processes defined to receive special scheduling treatment. Classes are defined by a system manager (or someone in authority), using the CLASP utility or a program that uses class-defining AOS/VS system calls.

Class is a property of a process, thus it exists only at runtime. It is formed by the process user locality (as specified to PREDITOR in a user's profile) and the program locality (as specified to SPRED in a program file). The intersection of the process's user locality and its program locality determines the process's class. You can define these two localities (called locality pairs) in the class matrix when you run CLASP.

A sample class matrix is shown in Figure Glossary-1, next.

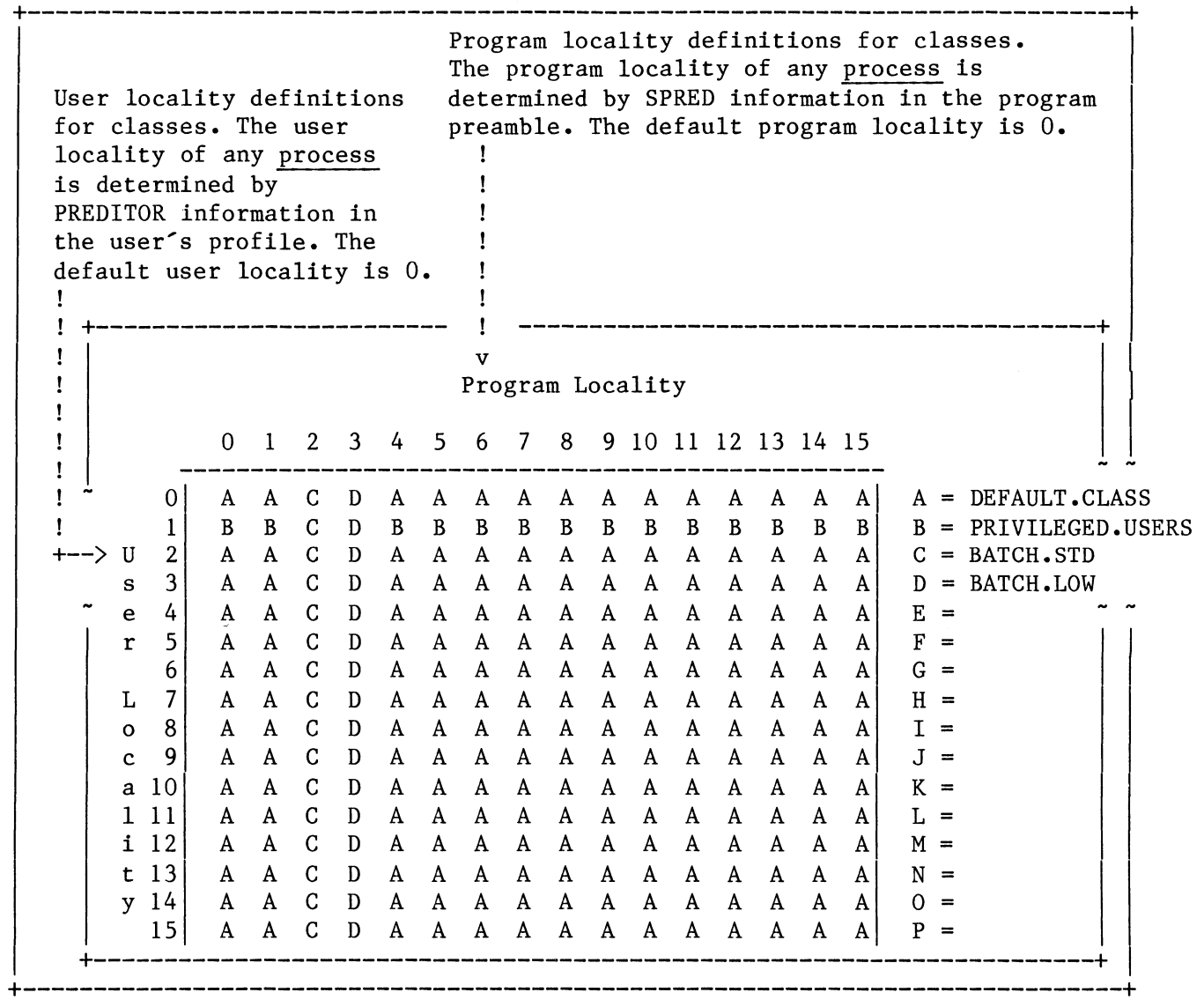

Figure Glossary-1. A Class Matrix Display

By default, there's one process class, class name key A, name DEFAULT.CLASS. You can create other classes, up to a total of 16. After creating classes, you'll want to allot processor time to each one -- using CLASP Main Menu choice 3 "Allot".

Figure Glossary-1 shows three classes in addition to the default class. Class PRIVILEGED.USERS includes all processes with user locality 1 (except those with program localities 2 and 3, which are defined as batch jobs). This privileged class is based entirely on the user profile; a PREDITOR run on any user's profile can allow that user to become privileged.

Class BATCH.STD includes all processes with program locality 2 and class BATCH.LOW includes all processes with program locality 3. Both batch classes are based entirely on programs; a SPRED run on any program file can make AOS/VS schedule the program as BATCH.STD or BATCH.LOW.

User localities can be changed at runtime (if the user has the privilege) via the LOCALITY command or PROCESS/LOCALITY switches.

Classes are not mandatory; you don't need to use them. The primary benefit of classes is control of job processor (CPU) time given to processes -- by user and/or by program. This control is useful in situations where

- you want to limit the amount of processor time given to a process or user (or set of processes/users); or
- you want to guarantee a process or user (or set of processes or users) a minimum amount of job processor time

without juggling user or process priority, type, and group. Class (and logical processor) names are maintained by CLASP, through files in the peripherals directory. AOS/VS knows classes and logicsl processors by ID -- a number from 0 through 15.

class matrix See class.

classless process A process that always runs in the default (mother) processor, regardless of the processor you specify. This pertains only in computers with more than one job processor. Any process that does system tasks (like the PMGR), and any process that defines a user device (issues ?IDEF), is a classless process. To identify classless processes, run PED and include the /MPROCESSOR switch; each classless process has a Y in its MP column.

compute bound A process that -- within a given interval -- issues relatively many job processor (CPU) instructions. The instructions might involve computations, or data comparisons and sorts. Compute-bound processes differ from I/O-bound processes (which demand a relatively large amount of I/O service. Multiple job processors (and classes and logical processors) are most helpful in systems that run compute-bound (not I/O bound) processes.

heuristic scheduling A form of scheduling in which the system bases a process's internal priority on its past behavior. Heuristic scheduling favors highly interactive processes; it penalizes compute-bound processes. In AOS/VS, all group 2 processes (group is based on process priority) are scheduled heuristically. By default, all user processes (and their sons) are group 2 processes.

A different form of scheduling is round-robin scheduling, in which each process gets the same amount of time (called a subslice) regardless of its past behavior.

job processor A hardware entity that computes and interprets program instructions. The term job processor includes and extends the standard definition of central processing unit (CPU). When AOS/VS starts up, it recognizes only the default processor, number 0. If your computer has more than one job processor, you must initialize the additional ones with the CLI command JPINITIALIZE. In a system with multiple job processors, the default processor (usually number 0) is called the mother. Each additional processor

is called a child.

I/O bound A process that demands more I/O than job processor attention. See also compute bound.

JP Abbreviation for job processor (CPU); see also job processor.

locality A number, or group of numbers, that determines the class of a process. There are two kinds of locality: user locality, (defined by PREDITOR in a user profile), and program locality (defined in a program file preamble by the SPRED utility).

With CLASP, you define locality pairs for each class. When a process runs, its user and program locality establish its class. A user may be able to change the user locality of a process (if his/her profile allows) using the CLI command LOCALITY or PROCESS command with /LOCALITY switch. The only way to change program locality is by editing the program file with the SPRED preamble editor, then running the program. See also class.

logical processor A scheduling structure you can create, allot classes processing time on, and connect to job processors -- via CLASP. A logical processor includes at least one class -- often several classes. You can create up to 15 logical processors with CLASP, and move job processors to different logical processor for different processing environments.

AOS/VS ships with a default logical processor named DEFAULT.LP, connected to the default job processor, number 0. If your computer has a second job processor and you bring it on line (JPINITIALIZE command), the system adds it to DEFAULT.LP; DEFAULT.LP then includes two job processors. (If you want any class other than the default class to get time on any logical processor, you must first create the desired class(es) and allot processor time to it (them) using CLASP.)

LP Abbreviation for logical processor; see also logical processor.

memory contention A state in which the system is rationing its physical (nondisk) memory. This occurs when all processes on the system ask for more physical memory that your computer has.

The system has two methods for handling memory contention: paging and swapping. In paging, the system takes a little-used page from a process, writes the page contents to disk, and gives the page frame to another process; later, if the original process needs the page, the system reads it back from disk. In swapping, the system takes all pages from a low-priority process, writes the contents to disk, and gives all page frames to another process. Disk access is much slower than memory access, thus paging and swapping resulting from memory contention can slow system response considerably.

The three major factors in system performance are processor usage, memory usage (amount of memory contention), and disk usage. CLASP can help with processor usage; but CLASP can't help detect or correct memory contention or disk bottlenecks. The PED program,

column BS (shows S for swapped), gives some idea of the amount of contention. (But for an accurate picture of memory contention you need the AOS/VS Performance Package). The DISCO program can give you information on disk usage.

mother processor The default job processor, number 0. If your computer has other job processors, each one is called a child processor. You can bring each child processor on line with the CLI command JPINITIALIZE.

primary class A class that's been allotted a specific percentage of processor time via CLASP. On any logical processor, primary classes get preference over secondary classes.

round-robin scheduling A form of scheduling in which each process gets the same amount of computer time (called a subslice) regardless of its past behavior. See also heuristic scheduling.

secondary class A class that's been allotted processor time by level, not percentage. See also primary class.

stranded process A process in a class whose logical processor has lost its last (or only) job processor. You can strand a process by moving its job processor to another logical processor or by creating a class environment in which the class can't be scheduled.

A stranded process acts as if it were blocked (which in fact it is, since it can't get processor time). If you want the process to continue, find a way to give it processor time -- usually by moving a job processor to a logical processor that can schedule the process's class; or by disabling class scheduling on a connected logical processor. Recovery procedures appear in Chapter 4.

System Manager privilege A special privilege, assigned by PREDITOR in a user's profile. In AOS/VS Revision 7.00, System Manager privilege allows a user to create and delete classes and logical processors, and to initialize and release job processors (JPINITIALIZE and JPRELEASE commands). It also allows a user to set the system date and time, start or stop system logging, and and change the system bias factor. System Manager privilege has no relationship to Superuser, Superprocess, or Access Devices privileges.

System Manager privilege can be turned on only by a system call, not by a CLI command. It can be turned on exclusively (which means no other process can turn it on until the original process turns it off) or nonexclusively. When you run CLASP without the /VIEW_ONLY switch, it turns on System Manager exclusively -- which means that only one CLASP process with the power to change AOS/VS can run at one time.

End of Glossary

Index

Within this index, the letter "f" means "and the following page"; "ff" means "and the following pages".

For each topic, primary page references are listed first.

V

W

X

Data General

TP_____

TIPS ORDER FORM
Technical Information & Publications Service

BILL TO:
COMPANY NAME_____
ADDRESS _____
CITY_____
STATE_____ ZIP _____
ATTN: _____

SHIP TO: (if different)
COMPANY NAME_____
ADDRESS _____
CITY_____
STATE_____ ZIP _____
ATTN: _____

QTY	MODEL #	DESCRIPTION	UNIT PRICE	LINE DISC	TOTAL PRICE

(Additional items can be included on second order form)

[Minimum order is $50.00]

Tax Exempt #_____
or Sales Tax (if applicable)

TOTAL	
Sales Tax	
Shipping	
TOTAL	

METHOD OF PAYMENT

☐ Check or money order enclosed
For orders less than $100.00

☐ Charge my ☐ Visa ☐ MasterCard
Acc't No._____ Expiration Date_____

☐ Purchase Order Number:_____

SHIP VIA

☐ DGC will select best way (U.P.S or Postal)

☐ Other:
☐ U.P.S. Blue Label
☐ Air Freight
☐ Other _____

NOTE: ORDERS LESS THAN $100, INCLUDE $5.00 FOR SHIPPING AND HANDLING.

Person to contact about this order _____ Phone _____ Extension _____

Mail Orders to:

Data General Corporation
Attn: Educational Services/TIPS F019
4400 Computer Drive
Westboro, MA 01580
Tel. (617) 366-8911 ext. 4032

Buyer's Authorized Signature Date
(agrees to terms & conditions on reverse side)

Title

DGC Sales Representative (If Known) Badge #

DISCOUNTS APPLY TO MAIL ORDERS ONLY

012-1780

educational services

CUT ALONG DOTTED LINE

DATA GENERAL CORPORATION
TECHNICAL INFORMATION AND PUBLICATIONS SERVICE
TERMS AND CONDITIONS

Data General Corporation ("DGC") provides its Technical Information and Publications Service (TIPS) solely in accordance with the following terms and conditions and more specifically to the Customer signing the Educational Services TIPS Order Form shown on the reverse hereof which is accepted by DGC.

1. **PRICES**
   Prices for DGC publications will be as stated in the Educational Services Literature Catalog in effect at the time DGC accepts Buyer's order or as specified on an authorized DGC quotation in force at the time of receipt by DGC of the Order Form shown on the reverse hereof. Prices are exclusive of all excise, sales, use or similar taxes and, therefore are subject to an increase equal in amount to any tax DGC may be required to collect or pay on the sale, license or delivery of the materials provided hereunder.

2. **PAYMENT**
   Terms are net cash on or prior to delivery except where satisfactory open account credit is established, in which case terms are net thirty (30) days from date of invoice.

3. **SHIPMENT**
   Shipment will be made F.O.B. Point of Origin. DGC normally ships either by UPS or U.S. Mail or other appropriate method depending upon weight, unless Customer designates a specific method and/or carrier on the Order Form. In any case, DGC assumes no liability with regard to loss, damage or delay during shipment.

4. **TERM**
   Upon execution by Buyer and acceptance by DGC, this agreement shall continue to remain in effect until terminated by either party upon thirty (30) days prior written notice. It is the intent of the parties to leave this Agreement in effect so that all subsequent orders for DGC publications will be governed by the terms and conditions of this Agreement.

5. **CUSTOMER CERTIFICATION**
   Customer hereby certifies that it is the owner or lessee of the DGC equipment and/or licensee/sub-licensee of the software which is the subject matter of the publication(s) ordered hereunder.

6. **DATA AND PROPRIETARY RIGHTS**
   Portions of the publications and materials supplied under this Agreement are proprietary and will be so marked. Customer shall abide by such markings. DGC retains for itself exclusively all proprietary rights (including manufacturing rights) in and to all designs, engineering details and other data pertaining to the products described in such publication. Licensed software materials are provided pursuant to the terms and conditions of the Program License Agreement (PLA) between the Customer and DGC and such PLA is made a part of and incorporated into this Agreement by reference. A copyright notice on any data by itself does not constitute or evidence a publication or public disclosure.

7. **DISCLAIMER OF WARRANTY**
   DGC MAKES NO WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY AND FITNESS FOR PARTICULAR PURPOSE ON ANY OF THE PUBLICATIONS SUPPLIED HEREUNDER.

8. **LIMITATIONS OF LIABILITY**
   IN NO EVENT SHALL DGC BE LIABLE FOR (I) ANY COSTS, DAMAGES OR EXPENSES ARISING OUT OF OR IN CONNECTION WITH ANY CLAIM BY ANY PERSON THAT USE OF THE PUBLICATION OF INFORMATION CONTAINED THEREIN INFRINGES ANY COPYRIGHT OR TRADE SECRET RIGHT OR (II) ANY INCIDENTIAL, SPECIAL, DIRECT OR CONSEQUENTIAL DAMAGES WHATSOEVER, INCLUDING BUT NOT LIMITED TO LOSS OF DATA, PROGRAMS OR LOST PROFITS.

9. **GENERAL**
   A valid contract binding upon DGC will come into being only at the time of DGC's acceptance of the referenced Educational Services Order Form. Such contract is governed by the laws of the Commonwealth of Massachusetts. Such contract is not assignable. These terms and conditions constitute the entire agreement between the parties with respect to the subject matter hereof and supersedes all prior oral or written communications, agreements and understandings. These terms and conditions shall prevail notwithstanding any different, conflicting or additional terms and conditions which may appear on any order submitted by Customer.

DISCOUNT SCHEDULES

DISCOUNTS APPLY TO MAIL ORDERS ONLY.

LINE ITEM DISCOUNT

5-14 manuals of the same part number - 20% 15 or more manuals of the same part number - 30%

DISCOUNTS APPLY TO PRICES SHOWN IN THE CURRENT TIPS CATALOG ONLY.

DataGeneral

TIPS ORDERING PROCEDURE:

Technical literature may be ordered through the Customer Education Service's Technical Information and Publications Service (TIPS).

1. Turn to the TIPS Order Form.
2. Fill in the requested information. If you need more space to list the items you are ordering, use an additional form. Transfer the subtotal from any additional sheet to the space marked "subtotal" on the form.
3. Do not forget to include your MAIL ORDER ONLY discount. (See discount schedules on the back of the TIPS Order Form.)
4. Total your order. (MINIMUM ORDER/CHARGE after discounts of $50.00.)

   If your order totals less than 100.00, enclose a certified check or money order for the total (include sales tax, or your tax exempt number, if applicable) plus $5.00 for shipping and handling.
5. Please indicate on the Order Form if you have any special shipping requirements. Unless specified, orders are normally shipped U.P.S.
6. Read carefully the terms and conditions of the TIPS program on the reverse side of the Order Form.
7. Sign on the line provided on the form and enclose with payment. Mail to:

   TIPS
   Educational Services – M.S. F019
   Data General Corporation
   4400 Computer Drive
   Westboro, MA 01580
8. We'll take care of the rest!

educational services

Data General Corporation, Westboro, MA 01580

093-000422-00