CSC 470: COMPUTER SYSTEM MODELLING AND ANALYSIS

MAP: MODELLING AND ANALYSIS PACKAGE

SOFTWARE SUMMARY

1.1 ACCESSING MAP

Your accounts will be set up so that MAP can be invoked by the command "map". To terminate a MAP session, type "quit" (or "exit y"). An alias in your .cshrc converts "map" into a reference to an executable file, and a path variable in your .cshrc will indicate the locations of standard files used through and by MAP.

1.2 INPUT FORMAT

MAP commands can be entered in free format fashion in either upper or lower case. Part of a command, a single command, or multiple commands can be entered on a single line. Comments can be entered anywhere that a blank could be reasonably typed, and can be extended over one or more lines. (Comments help annotate MAP runs and MAP EXECs.) Comments are delimited by double braces, as in {{comment line}}, or by single braces, as in {comment line}. The former type of comment is copied to the currently opened output file (including the terminal), while the latter is not.

1.3 ABBREVIATIONS

Commands and object names can be abbreviated to the shortest unique abbreviation for their context. Thus, CEN CP is equivalent to CENTER CPU, if there is no center (other than CPU) whose name begins with CP. If the name of one center, for instance, is a prefix of the name of some other center, and the shorter name is typed in full, it is assumed that the user means the center whose name matches exactly. Thus, CEN CP sets the center to CP rather than to CPU if both centers exist.

It is possible to specify that a name should not be taken as an abbreviation by appending a period (‘.’) to the name. This can be useful when creating a new object with a name that is an abbreviation of an existing object of that type. For instance, if the CENTER named CPU already exists, the command CENTER CP will set the current center to CPU. In contrast, the command CENTER CP. will create a new center with name CP.

Some names cannot be abbreviated. File names (for instance, as used by SAVE, READ, and EXECs) must be typed in full. Similarly, the keywords EACH, EVERY, and CANCEL cannot be abbreviated.

1.4 CONTEXT

Most MAP commands serve either to define or alter the MAP model, or to request output performance measures of the current model. For each component type in a MAP model (class, center, domain, string, controller, channel, node, and link) there is at all times a ‘‘current context’’ to which commands refer implicitly. The current context for each component type is given by the command with the same name as the component type. For instance, classes, centers, and domains are made current by issuing an appropriate CLASS, CENTER, or DOMAIN command.

The purpose of context is to reduce the amount of typing required for command entry. By maintaining an implicit context, the user can enter several commands for the current context without having to specify, for instance, the class, center, or domain affected with each command. For example, in a model with class PRODUCTION current, the commands AVG_MPL 10 INDOMAIN PROD would set the multiprogramming level of PRODUCTION to 10 and place it in domain PROD. Notice that neither

command required the class name (PRODUCTION) as an explicit parameter.

It is possible to make the current class, center, or domain either EACH or EVERY. Context EACH causes MAP to prompt for parameters for each entity of the appropriate type. Context EVERY causes the (single) parameter given to be applied to every entity of the appropriate type. For example, suppose that a model contains three classes, named TSO_1, TSO_2, and BATCH. If the current class context is EACH, the command

```
ARRIVAL_RATE    .7
```

sets the arrival rate of class TSO_1 to .7, and causes MAP to prompt for the arrival rates of classes TSO_2 and BATCH. If the current class context is EVERY, the same command sets the arrival rate of all three classes to .7 without the need for further input.

MAP also allows the user to make arbitrary groups of components the current context. For instance, suppose a model contains a total of ten classes with names TSO_1, TSO_2, ..., TSO_5, BATCH_1, BATCH_2, ..., BATCH_5. The commands

```
CLASS    TSO_3
CLASS    +TSO_5
CLASS    +BATCH_3
```

set the current class context to {TSO_3, TSO_5, BATCH_3}. The command

```
PLEVEL    4
```

will now set the priority level of all three classes (and no others) to 4. (Thus, a group defined in this way acts like EVERY with respect to how input is treated.) The first command makes class TSO_3 the current class context. The '+' prefix in the parameters of the second and third commands cause classes TSO_5 and BATCH_3 to be added to the current context, rather than to replace it. Prefix '-' removes objects from the current context. For example, after executing the three commands above the command

```
CLASS    -TSO_5
```

would leave the class context as {TSO_3, BATCH_3}.

There is a more convenient way to define groups when names share a common prefix. For instance, classes TSO_1 through TSO_5 could be made the current context by the single command

```
CLASS    TSO*
```

The appended '*' causes MAP to match all classes whose names begin with 'TSO'. The '*' suffix can be combined with prefix operators. For instance,

```
CLASS    +BATCH*
```

would cause the context to be all ten classes (and thus semantically equivalent to EVERY). These two commands together have the same effect as

```
CLASS    *
```

It is possible to give a name to the current context for a component type when the context forms a group. For instance,

```
CENTER     DISK*
GDEFINE    DASD    CENTER
```

makes the current context the group of centers whose names begin with 'DISK', e.g., {DISK1, DISK2, DISK3}. The GDEFINE command defines a group whose name is DASD and whose value is given by the current center context. The name of this group, i.e., 'DASD', can now be used as a shorthand in subsequent CENTER commands to make the set of centers {DISK1, DISK2, DISK3} the current context. That is,

```
CENTER    DASD
```

is valid and means

```
CENTER    DISK1
CENTER    +DISK2
CENTER    +DISK3
```

Notice that the group represents the specific centers in the current context at the time of the GDEFINE command, rather than the pattern ('DASD*') used to set up that context. Thus, if a new center, DISK4, were created between the GDEFINE command and subsequent use of the name DASD in a CENTER command, DISK4 would not be made part of the current context.

Group names may be abbreviated. However, any parameter to the CENTER command (for instance) will be matched against valid center names before any attempt is made to match against center group names. Thus if there is a center named DASD_FARM the command above will refer to this one center alone rather than the group. (In this case the desired effect can be obtained using 'CENTER DASD.'.)

Immediately after the execution of a CANCEL command, there is no context for the entity of the type canceled. (For instance, CANCEL CENTER removes the current center from the model and results in there being no current center context.) Commands that require context will cause errors if no context exists when the commands are executed. MAP uses the keyword NONE on output to indicate that no context exists. NONE has no special meaning on input.

1.5 CREATING MODEL COMPONENTS

Classes, centers, and domains are created by the CLASS, CENTER, and DOMAIN commands, respectively. Whenever one of these commands refers to an object not currently defined in the MAP model, the object is created with the given name. Thus, CLASS PGROUP1 creates a new class with name PGROUP1 if no class with that name (as a prefix) currently exists.

(To define a new center, say, whose name is a prefix of an existing center, append a period (.) to the new name. This indicates that the name has been typed in full, and should not be considered as a possible prefix of some other name.)

1.6 SPECIFYING PARAMETERS

Parameters to MAP commands consist primarily of numeric values (real and integer values) and character string names. These names can be any character string of 12 characters or less. (Names longer than 12 characters are truncated.) There are two keywords that cannot be used as names: EACH and EVERY. These have special meanings, as noted in paragraph 2.3.

When a command has been issued that causes MAP to prompt (usually because the context of the relevant entity is EACH), typing a null line in response to the prompt causes MAP to proceed to values for the next class (if any) without affecting intervening values.

Numeric valued parameters can be specified in any of three ways: a numeric literal (e.g., 2.7), an expression (e.g., (3*2.184) / (17.7+3.8*2.2)), or an operator suffixed value (e.g., 2*).

Numeric literals are specified in fixed format, and cannot be given in scientific notation. For example, 127.5 is valid but 1.275E02 is not.

Expressions can include parentheses and MAP EXEC variables (see Chapter 3 of this manual), but may not contain embedded blanks. They are evaluated using standard operator precedence rules: parenthetical subexpressions are evaluated first, unary operators (+ and -) have the next highest precedence, multiplication (*) and division (/) come next, and finally addition (+) and subtraction (-). Equal precedence operators are evaluated left to right. For example, the value of 5*6-6/3 is 28, while the value of 5*(6-6)/3 is 0.

Unary '@' is absolute value, that is, that magnitude of the operand is preserved but the sign is made positive. Unary '-' is negation, that is, the sign of the operand is reversed. The result of a unary '-' can therefore be either positive or negative. A unary operator precedes the operand it operates on. For example, '@(&F1-&F2)/&F2*100' gives the percentage difference between F1 and F2 and forces the result to be positive. A result could be forced to be negative (something that is not normally needed) by negating the absolute value of the result, e.g., '-@(&F1-&F2)/&F2*100' will be a negative percentage

difference.

Operator suffixed values serve as modifiers to the current value, rather than as replacements. Thus, ARRIVAL 2* doubles the arrival rate of the current class, rather than setting it to 2 (as ARRIVAL 2 would). The value suffixed can be either a literal value (e.g., 2*) or an expression (e.g., 1+1*).

Some MAP commands take integers as parameters. In these cases, any real value given is truncated to an integer. For example, PLEVEL 2+3/4 results in priority level 2 being assigned to the current class.

SELECTED COMMANDS

This chapter lists the MAP commands and their effects. Examples of each command are given. These examples might contain multiple MAP commands on a single input line or use command name abbreviations to show these features in the MAP input language.

1. ARRIVAL_RATE

Command:	ARRIVAL_RATE
Parameter:	Real valued arrival rate.
Effect:	If the current class is not EACH, the arrival rate of the current class is set by the parameter. (If the current class is EVERY, the arrival rate of every class with either transaction or undefined type is set to the single parameter value given.) If the current class is EACH, MAP prompts for values for each class with undefined or TRANSACTION type. If the type of a current class is undefined, it is made TRANSACTION.
Errors:	Current class is not EACH or EVERY and is not TRANSACTION type.
	No current class.
	Negative arrival rate.
Example:	ARR 6+
	The arrival rate of current class is incremented by 6.

2. AVG_MPL

Command:	AVG_MPL
Parameter:	Real-valued average multiprogramming level.
Effect:	If the current class is not EACH, the multiprogramming level of the current class is set to the parameter value. (If the current class is EVERY, all classes with BATCH or undefined type are set.) If the current class is EACH, MAP prompts for values for each class of undefined or BATCH type. If the type of the current class is undefined, it is made BATCH.
Errors:	Current class is not EACH or EVERY and is not BATCH type.
	No current class.
	Negative average multiprogramming level.
Example:	CLA PROD AVG 8
	The multiprogramming level of PROD is set to 8.

3. CANCEL

Command:	CANCEL
Parameters:	ALL, CENTER, CHANNEL, CLASS, CONTROLLER, DOMAIN, LINK, NODE, STRING, or none.
Effect:	If a parameter is given, the currently defined object of that type is removed from the model and the current object of that type is set to NONE. If no parameter is given, or the parameter ALL is given, all objects in the model are canceled, giving a null model. In each case, the action must be confirmed by answering Y or N to the appropriate prompt.
Errors:	Ambiguous or unknown parameter. Channels and controllers cannot be canceled if currently part of a logical channel.
Examples:	CAN CLASS Assuming the current class was IMS, MAP would respond with CANCEL CLASS IMS (Y OR N)?. N (No) terminates the command. Y (Yes) results in the removal of the IMS job class from the model. CAN MAP asks for confirmation to cancel the entire model and then acts according to the response.

4. CENTER

Command:	CENTER
Parameters:	The name of a service center, EACH, EVERY, or none
Effect:	If no parameter is given, the name of the current center is printed. If EACH or EVERY is given, subsequent commands affecting service centers refer to all currently defined service centers. If a service center name is given, that center is made the current center. If a center with the name given does not currently exist, a new center is created with that name. New centers are initialized so that the loadings of all classes at them are 0.0 and the scheduling disciplines are NORMAL.
Errors:	An attempt is made to create a new center when the model already contains the maximum number of centers allowable. EACH and EVERY are not allowed when the current model contains no centers. Ambiguous center name.
Examples:	CEN CP Assuming center CPU was the only center whose name began with CP, it would be made the current center. Subsequent commands affecting service centers refer implicitly to that center (until the next CENTER command). CEN DISK3 If no center had a name beginning with DISK3, MAP would respond that a new service center had been created (with name DISK3). The LOADINGs of all job classes at DISK3 would be set to 0.0. DISK3 would have NORMAL scheduling.

5. CLASS

Command:	CLASS
Parameters:	The name of a customer class, EACH, EVERY, or none.
Effect:	If no parameter is given, the name of the current class is printed. If EACH or EVERY is given, subsequent commands affecting customer classes refer to all classes. If a class name is given, that class is made the current class. If a class with the name given does not currently exist, a new class is created with that name. New classes are initialized to have loadings of 0.0 at all service centers. All other parameters are undefined.
Errors:	An attempt to create a new class is made when the current model possesses the maximum number of classes allowable. EACH and EVERY are not allowed when the model contains no classes. Ambiguous class name.
Examples:	CLA MAP responds by printing the name of the current class. CLA TSO The current class is set to TSO. CLA NEWIMS A new class (NEWIMS) is created, with LOADINGs of 0.0 at all service centers. The workload intensity and type of NEWIMS are undefined.

6. CLOSE

Command:	CLOSE
Parameters:	None.
Effect:	The currently open output file (see OPEN) is closed and removed from the stack of output files. Thus, the previous most recently opened file becomes the output report destination file.
Error:	No currently open file.
Example:	CLOSE The current output file (if any) is closed.

7. EXIT

Command:	EXIT
Parameters:	None.
Effect:	Terminate this MAP session. If the model current at the time does not exist as a permanent MAP data file, a warning is printed. The user is requested to confirm his intention to exit the MAP session by typing a Y.
Errors:	None.
Example:	EXIT
	The user is asked to confirm his intention to exit, and upon doing so the MAP session is ended.

8. HELP

Command:	HELP
Parameters:	ALL, CENTER, CHANNEL, CLASS, CONTROLLER, DOMAIN, EXECS, FILES, GRAPHS, IO, MEMORY, REPORTS, STRING, or none.
Effect:	If no parameter is given, a list of valid parameters is printed. Otherwise, a list of command names and their effects is printed. The parameter determines which subset of commands is listed. If the parameter is ALL, all commands are listed. In other cases, only those commands concerned with the named subject type are listed.
Errors:	None.
	Invalid or ambiguous parameter.
Example:	HELP ALL
	A summary list of MAP commands is printed at the terminal.

9. JOURNAL

Command:	JOURNAL
Parameters:	ON, OFF, ALL, or USER.
Effect:	Journaling allows the user to capture a MAP session in a log file. (Appendix D or E has the name of the log file on the user's system.)JOURNAL ON captures an exact copy of what appears on the user's terminal. JOURNAL ALL captures all input and output lines, including those read by READ commands and by the execution of MAP EXECs. JOURNAL USER captures only those lines entered at the terminal by the user, thus creating a session log that can be replayed through MAP at a later time. JOURNAL OFF turns off journaling. If journaling is turned ON (ON, ALL, or USER) and OFF repeatedly during a single MAP session, the JOURNAL output is appended in a single file.
Errors:	Error in attempt to open the JOURNAL output file. Invalid parameter.
Example:	JOURNAL ON Each line written to the terminal by MAP, or entered at the keyboard by the user, is copied into the JOURNAL file.

10. LIST

Command: LIST

Parameters: ALL, CENTER, CHANNEL, CLASS, CONTROLLER, DOMAIN, GROUPS, IOCONFIGURATION, LOADINGS, MEMORY, STRING, or none.

Effects: A description of the currently defined model is printed at the currently opened output destination (by default, the terminal). If no parameter is given, the entire model is listed. Otherwise, only the portion of the model corresponding to the parameter is printed. For IOCONFIGURATION, that portion includes SETSYSTEM if ON, and information on CHANNELs, CONTROLLERs, and STRINGs. For MEMORY, the information printed is SETSYSTEM if ON, MEMSIZE if a value has been specified, PAGELOADs if any have been given, and SWAPLOADs if any have been given.

The information printed for each model component is:

CLASSES - Name, TYPE, THINK, workload intensity, PLEVEL, %PLEVEL, DPLEVEL, INDOMAIN, NON_DISPATCHABLE, PAGECOUNT, and SWAPCOUNT of the current class(es). (An asterisk following the PLEVEL indicates an active PLEVEL distribution for that class. The distribution appears at the end of the class information.) To avoid confusing inexperienced users, headers for some fields are suppressed if no user data for that field has been given. In particular, PAGECOUNT and SWAPCOUNT fields cannot appear in the report.

CENTERS - Name, SCHEDULE, PROCESSORS, MODEL, ONSTRING, IOTIME, CV2, UNIT, BLOCKSIZE, %SEEK SHARED_USE, RHIT, WHIT, RW_RATIO, and CACHING. BLOCKSIZE, CONNECTTIME, ONSTRING, IOTIME, %SEEK, SHARED_USE, RHIT, WHIT, CACHING, and RW_RATIO fields are suppressed if no user data for them has been given. CV2 is printed only if some center has a CV2 not equal to 1.0. Only one of BLOCKSIZE and CONNECTTIME is ever printed. If user data for CONNECTTIME has been given, that field is printed. The user should not use both BLOCKSIZE and CONNECTTIME in a single model.

CHANNELS - Name.

CONTROLLERS - Name, ONCHANNEL, and MODEL.

DOMAINS - Name, LIMITED_MPL, ADMIT, and classes in this domain (ordered highest to lowest priority, if appropriate).

GROUPS - Names and members of currently defined CLASS, CENTER, DOMAIN, CHANNEL, CONTROLLER, STRING, LINK, and NODE groups. (See Section 2.4.)

IOCONFIGURATION - Information on STRINGS, CONTROLLERS, and CHANNELS.

LOADINGS - The loadings of the current class(es) and center(s).

MEMORY - MEMSIZE, SETSYSTEM if ON, PAGELOADs, and SWAPLOADs.

STRING - Name, ONCONTROLLER, and MODEL.

Errors: CENTER, CHANNEL, CLASS, CONTROLLER, DOMAIN, or STRING is specified and there is no current center, channel, class, controller, domain, or string, respectively.

Unknown or ambiguous parameter.

Examples: LIST

A description of the current model is printed at the terminal.

LIST CONT

A description of the current controller is displayed at the terminal.

Note: The OPEN command can direct the output report to a file.

11. LOADING

Command: LOADING

Parameter: Device loading(s).

Effects: The total service time requirement of the customers in the current class at the current center is given by the parameter. If either the current center or the current class is EACH, MAP prompts for the appropriate data. If either the current center or the current class is EVERY, the parameter value(s) given apply to every entity of that type.

Loadings should typically be computed from measurement data as:

(Utilization x measurement interval) / (number of completions)

The LOADING should not include any busy time due to SHARED_USE. The LOADING should include PAGELOAD and SWAPLOAD times.

Errors: No current class.

No current center.

The LOADING of a class at a center must be greater than PAGELOAD + SWAPLOAD) * 1.01. (The multiplicative factor is to allow for roundoff errors when setting the values of these parameters.) This error is detected only when a performance report is requested.

The loading given is negative.

Examples: CLASS TSO CENTER CPU LOAD 2.1

The total service requirement of class TSO at center CPU is set to 2.1.

CLASS EVERY CENTER CPU LOAD 2/

The loading of every class at center CPU is halved.

12. MODEL

Command: MODEL

Parameters: A character string, device model, or none.

Effects: Models Models can be given to centers (including CPU and DASD centers), STRINGs, and CONTROLLERs. The model of a CENTER is simply a character string indicating a manufacturer designation (e.g., 5880 or IBM3350). The model of a STRING or CONTROLLER is the name of a TDEFINE-defined string model. The model specification allows the user to indicate in listings the model of each service center. Additionally, some models are known to MAP, and are used in automatically altering parameters when the model of a center is changed during modification analysis use of the model (i.e., when SETSYSTEM is ON). (See also the MOVE command for another use of the MODEL information.) If SETSYSTEM is ON, when the model of a service center, STRING, or CONTROLLER with a model known to MAP is changed, MAP automatically updates the MAP model parameters to reflect the change If SETSYSTEM is OFF, or a model is not known, MAP prints a warning and leaves the parameters unchanged.

When a model parameter is given with the MODEL command, MAP first attempts to determine if it is a CPU, DASD, or STRING model. If the model is either CPU or DASD, or is not known to MAP, the current center is assigned the named model. If the model is a STRING model, the current string is assigned the model If the model is a CONTROLLER model, the current controller is assigned the model.

If no parameter is given, MAP prints the names of all known models.

Errors: No current center/string/controller.

Current center/string is EACH.

Example: CEN DISK2 MODEL IBM3350

DISK2 is converted into an IBM3350 model disk. The loadings of all classes at DISK2 are automatically changed by MAP to reflect the change in device model if the previous model of DISK2 was known.

Notes: Models can be created and canceled, and their parameters examined and modified with the TDEFINE command.

When the model of a DASD center is changed and SETSYSTEM is ON, some model parameters might not be fully updated until an output report command (e.g., PERFORMANCE) is issued.

The file INITDEVS MAPEXEC contains the definitions for the default models, and is read automatically by MAP at the start of each session. The file USERDEVS MAPEXEC can augment the list of default MODEL definitions.

13. ONLINE_USERS

Command:	ONLINE_USERS
Parameter:	Number of online users.
Effect:	The average number of online users for the current class is set by the parameter. If the current class is EACH, MAP issues prompts for each class with undefined or TERMINAL type. If the current class is EVERY, the single parameter value is applied to every class. If a current class had undefined type, it is made TERMINAL type.
Errors:	No current class.
	The current class is not EACH or EVERY and is not TERMINAL type.
	Number of online users is less than zero.
Example:	CLA TSO ONLINE 15+
	The number of TSO terminal users is increased by 15.

14. OPEN

Command:	OPEN
Parameters:	A filename, TERMINAL, or none.
Effects:	If a filename is given, the file is opened and the outputs of subsequent report generating commands (IOREPORT, MEMREPORT, PATHREPORT, PERFORMANCE, QLENGTHS, RESIDENCE, and UTILIZATION), as well as that of the LIST and TITLE commands, are printed in this file rather than at the terminal. Report outputs are appended to the file until a CLOSE command is issued to close the file. Additionally, all comments enclosed by double braces, as in {{comment}}, and the output of PLOT, WRITE, and WRITELN commands are copied into the output file. If TERMINAL is given, output is directed to the terminal. If no filename is given, the name of the current output file is printed at the terminal. Open files are kept on a stack so that more than one file can be OPEN at a time. However, only the most recently opened file is active, and all report output is sent to it.
Error:	Maximum number of output files (currently four) already open.
Examples:	OPEN NEWOUT Reports and comments are sent to file NEWOUT. OPEN NEWOUT2 NEWOUT is suspended and NEWOUT2 is made the active file.
Note:	Output files are stored as files with the same filename as the MAP filename, and a filetype that indicates the file is a MAP output file. Consult Appendix D to this manual. OPEN TERMINAL should be used to temporarily redirect output to the terminal while creating an output file, since closing and then opening the file will cause previous output to be lost. The CLOSE command should then be used to terminate outputting to the terminal and to resume appending output to the file.
CAUTION:	In some file systems, having multiple simultaneously opened files can cause the destruction of files. Consult Appendix E to this manual.

15. PERFORMANCE

Command:	PERFORMANCE
Parameters:	None.
Effect:	The major performance measures of the current model are printed. For each class, the time in system, memory queuing time, terminal network delay, system response time (time in system plus memory queuing time), throughput, CPU utilization , and expansion factor (ratio of response time to the sum over all centers of the LOADINGs) are given.
Error:	The model must be well-formed, i.e., all necessary parameter values must have been previously specified. If not, an error message is printed indicating that a value is missing.
Example:	PERF
	System performance measures for each class in the model are printed.
Notes:	The OUTPUT command can control the entries in the report.
	Command OPEN can direct the report to a file.

16. PLEVEL

Command:	PLEVEL
Parameter:	An integer priority level.
Effect:	The dispatching priority of the current class at priority scheduled service centers (if any) is set to the integer parameter. Higher priority numbers indicate preemptive priority over lower priority numbers. If the current class is EACH, MAP prompts for priority levels for each class. If the current class is EVERY, the priority level of every class is set to the single parameter value given. MAP allows each class to be assigned a *priority level distribution*. A distribution consists of two or more PLEVELs, each of which applies for a specific fraction of the loadings required at the priority centers. (These fractions are specified using the %PLEVEL command.) When output reports are requested, the sum of the fractions over all PLEVELs must be 1.0 for each class. To simplify maintaining this property, MAP assigns a %PLEVEL to each new PLEVEL created for a class that equals 1.0 minus the sum of the fractions associated with the current PLEVELs. For instance, CLASS TSO PLEVEL 4 %PLEVEL .3 PLEVEL 2 creates a PLEVEL of 4 and automatically gives it a %PLEVEL of 1.0. The explicit %PLEVEL command then assigns a fraction of .3 to that PLEVEL. PLEVEL 2 is then created, and automatically assigned a %PLEVEL of .7. %PLEVEL always affects the last PLEVEL mentioned in a PLEVEL command for the current class. A %PLEVEL 0 command removes a PLEVEL from the current class.
Errors:	No current class. A zero or negative priority level is given.
Example:	PLEVEL 5 The current class is given a priority level of 5 at the priority scheduled service centers (if any).

17. PROCESSORS

Command:	PROCESSORS
Parameter:	An integer number of processors from 1-999.
Effect:	The parameter indicates the number of processors to be assigned the current center. Each processor can serve a single customer, so a center with N processors can serve up to N of the customers in its queue simultaneously.
Errors:	The number of processors must be in the range 1-999.
	No current center.
Example:	CEN CPU PROC 2
	The CPU is defined as a (tightly coupled) dual processor.
Notes:	The EFFECTIVE_RATES parameter of TDEFINE can specify the effective aggregate rate at which the processors deliver service as a function of the number of jobs at the processor.
	If the center does not have a MODEL with defined EFFECTIVE_RATES, the rate with N customers in the queue defaults to 1.0+0.8(N-1), i.e., each additional processor beyond the first contributes 0.8 of a full processor.
	If the LOADING of a multiprocessor is measured from an existing system, a SETCPU command should be issued after specifying the LOADINGs of all classes at the CPU.

18. QLENGTHS

Command:	QLENGTHS
Parameters:	None.
Effect:	Prints a report of mean queue lengths resulting from the current model at all service centers and memory for each class. Queue lengths include both jobs waiting and jobs in service. To compute the average number of waiting jobs, subtract the utilization (see UTILIZATION) from the queue length.
Error:	If the model is not defined completely, an error message is printed.
Example:	QLEN
	The queue lengths for all classes at all service centers (in the current model) are printed at the terminal.
Notes:	The entries in the output report can be controlled by the OUTPUT command.
	The OPEN command can direct the report to a file.

19. READ

Command:	READ
Parameter:	The name of a MAP model definition file name.
Effect:	The MAP model stored in the external file is made current, replacing the current model. If the current model has not been saved in a MAP model definition file, MAP prompts for confirmation of the read, since the current model will be lost.
Errors:	If the file does not exist, an error message is printed.
	A file created by SAVE might contain errors if some parameters to be saved were too large for the field used to write them. The error in the existing saved model file can be corrected only by editing it. Typically, a string of asterisks is printed by SAVE in place of the required (too large) parameter. The more permanent solution to this situation is to change the time scale of the model to reduce the size of the parameters.
	The file is presumed to contain valid MAP commands. Invalid commands will cause appropriate error messages.
	The attempt to open the data file causes the MAP internal limit on the number of open input files to be exceeded.
Example:	READ AMDAHLV
	MAP asks for confirmation of the read, since the current model will be overwritten. Upon confirmation, the MAP model stored in file AMDAHLV is made current.
Note:	The named file must be stored with a file type that indicates a MAP data file.

20. RESIDENCE

Command:	RESIDENCE
Parameters:	None.
Effect:	The residence time in the current model of each class at every center and memory is printed. Residence time is the sum of queuing and service times.
Error:	If the model is not completely defined, an appropriate error message is printed.
Example:	RES
	The average amount of time spent at each center (and domain) by each class in the current model is printed at the terminal.
Notes:	The entries in the output report can be controlled using the OUTPUT command.
	The OPEN command can direct the report to a file.

21. SAVE

Command: SAVE

Parameter: A MAP model definition file name.

Effect: The current model is saved in a MAP model definition file with the name given by the parameter. The file consists of MAP commands, and can be edited by editors external to MAP. If the file already exists, MAP requests confirmation that the file should be replaced.

Error: The attempt to open the output file causes MAP to exceed its internal limit on the number of open output files (see OPEN).

Example: SAVE AMDAHLV

The current model is saved in file AMDAHLV.

Notes: The file created has the name given as the parameter to the command and a file type that indicates a MAP data file. See the appropriate appendix at the end of this manual.

If a time unit other than seconds has been used to construct the model, or if the model contains some very large or very small parameter settings, it is possible that SAVE will create a file that cannot be read successfully by READ. If numeric parameter values too large to be written in the number of output columns allocated by SAVE are encountered, a string of asterisks is written in that field. Since asterisks are not valid numeric parameters, when READ encounters this data an error occurs. The solution at that point is to edit the MAPDAT file and replace the asterisks with a valid parameter. If very small parameter values exist, the field used to write them might not be large enough to exhibit any significant digits. In this case, the parameter is written as zero. This can sometimes cause unexpected results after the file is READ. Both these problems are undetected until the saved file is READ.

SAVE makes use of two commands intended for its use only: AUTHOR and !RPS. These should not be edited by the user. If a center is added or deleted from the stored model file by an editor, the parameters to !RPS might have to be modified. (See the description of the !RPS command.) Editing of saved files is not recommended for this reason.

22. SCHEDULE

Command:	SCHEDULE
Parameters:	DELAY, NORMAL, or PRIORITY.
Effect:	The scheduling discipline at the current center is set to DELAY, NORMAL, or PRIORITY, as appropriate. At DELAY service centers, all customers are served in parallel at full rate (no queuing occurs). At NORMAL centers, queuing occurs in either FCFS or round-robin fashion. At PRIORITY centers, priority scheduling is used. (Class priority values are set by the PLEVEL command.)
Errors:	No current center.
Example:	SCHED NORMAL
	The CPU scheduling discipline is set to NORMAL.

23. THINK

Command:	THINK
Parameter:	Real-valued think time.
Effect:	The parameter is taken as the think time of the current class. If the current class is EACH, MAP prompts for think times for all classes with undefined or TERMINAL type. If the current class is EVERY, the think times of all classes with undefined or TERMINAL type are set by the single parameter value given. If a current class has undefined type, it is made TERMINAL.
Errors:	No current class.
	The current class is not EACH or EVERY and the type of the current class is not TERMINAL.
	The think time given is not positive.
Example:	CLASS TSO THINK 15
	The think time for TSO users is changed from 10 to 15 seconds.

24. TYPE

Command:	TYPE
Parameters:	BATCH, TERMINAL, TRANSACTION, or none.
Effect:	The type of the current class is set to the given parameter. If the current class is EACH, MAP prompts for a type for each class. If the current class is EVERY, all classes with undefined types are set. The average multiprogramming levels are undefined. If no parameter is given, the type of the current class is displayed.
Error:	No current class.
	Unknown or ambiguous parameter.
Example:	CLA TSO TYPE TRAN
	TSO class is changed to TRANSACTION type. The average arrival rate is undefined.

25. UTILIZATIONS

Command:	UTILIZATIONS
Parameters:	None.
Effect:	The utilizations of each service center by each class for the current model are printed. For centers with more than one processor, the average utilization of the processors is given. (Utilizations range from 0.0 to 1.0.)
Error:	If the current model is not completely defined, an error message is printed.
Example:	UTIL
	The utilizations of the service centers are reported.
Notes:	The OUTPUT command can control the entries in the output report.
	The OPEN command can direct the report to a file.
	The TOTAL utilization field for each center does not include a SHARED_USE utilization.

26. ?

Command:	?
Parameters:	None.
Effect:	The names of the current class, center, domain, channel, controller, and string are printed.
Errors:	None.
Example:	?
	MAP responds with the names of the current class, center, domain, channel, controller, string, node, and link. All but the first three are surprised if no entities of that type have been defined in the current model.