/

PAWAN LAWALE

Phase III (Due April 22)

A MULTIPROGRAMMING BATCH SYSTEM: In Phase II of the project, a memory management scheme that can support multiprogramming was developed. In Phase III, the goal is to introduce multiprogramming. To do so, a process manager needs to be implemented which shall have Round Robin scheduling. In what follows, the terms job and process are used interchangeably.

PROCESS MANAGER

For every job in the SYSTEM, the Process Control/Context Block (PCB), which was outlined in Phase II, needs to be extended. It will now include at least the following information:

a) registers (PC, IR, TOS)
b) time the job entered the SYSTEM
c) cumulative execution time (i.e., CPU time) for the job
d) the expected completion time of the current I/O operation of the job – used in transfering job from blocked to ready

Once in the SYSTEM, each job may be in one of the following states:

running: A job is running when it is on the CPU (executing). Processes will be given time slices or quantums of 20 virtual time units to run.

ready: A job is ready when it is waiting to be dispatched. Ready processes are kept in the ready queue or ready list.

blocked: A process is blocked when it is waiting for a page transfer completion (this includes I/O operations, as explained below). Blocked processes are kept in the blocked queue (list), which is organized in ascending transfer time to the ready queue (based on the completion time of the current I/O operation).

21

Initially, as many jobs as would fit in memory are spooled onto the DISK and the first page of the first job is loaded from the DISK into a memory page frame. Here "first page" means "the respective job page pointed to by the initial value of the job's PC", which is the way it was in Phase-2.

Execution then starts and continues until the job exits the CPU. Whenever a process exits the CPU, an appropriate action has to be taken (see below), and another job has to be found to run. First, jobs on the blocked queue that have become ready are moved to the end of the ready queue. Second, if there is enough memory for other jobs, new jobs are started (spooled in). And third, the ready queue is serviced. The spooling of jobs to the DISK takes place instantaneously taking zero virtual time units (as in Phase II), and a page transfer to main memory now takes 20 time units.

Each job gets a time quantum of 20 virtual time units each time the CPU is allocated to it. Once a job is running, one of six events could occur.

1) If the job uses all of the 20 virtual time units and still needs more CPU time, it loses the CPU and control is transferred to the SYSTEM. The SYSTEM will be responsible for appending the job to the ready queue.

2) If the job terminates before its time quantum expires, the CPU is released and control is transferred to the SYSTEM which is responsible for destroying the process, releasing the allocated memory, updating the relevant tables, spooling out the job, and releasing the allocated DISK space. As in Phase II, the output spooling takes zero virtual time units, i.e., the CLOCK is not incremented.

Upon termination of each job, the following information is to be output (to the execution_profile, see below):

a) internally-assigned Job Id
b) warning/error messages (if any)
c) Input-Data Segment, labeled by the respective Job Id
d) Output-Data Segment, labeled by the respective Job Id (if no output was generated, your SYSTEM will print a message to that effect)
e) A message indicating the nature of termination (normal or abnormal, plus a descriptive error message if abnormal)
f) time (clock reading) the job entered the SYSTEM in HEX [arrival time]
g) time (clock reading) the job is leaving the SYSTEM in HEX [departure time]
h) number of page faults generated by the job

i) run time for the job (in DECIMAL) subdivided into execution time, I/O time, page-fault handling time, and segment-fault handling time
j) the turn-around time (departure time minus arrival time) in DECIMAL

3) If a job exits the CPU because of an I/O operation (if it is the first time, see Item 6 below), control is transferred to the SYSTEM. If the referenced page is in the main memory, the process will be appended to the ready queue (with no additional time increment required since what the job needs is already already in the memory), otherwise it will be appended to the blocked queue.

(Reminder: I/O for Phase III is as in Phase II, i.e., disk I/O by paging, and not as in Phase I, i.e., I/O via keyboard and screen.)

4) If a run-time error (a fatal error and not a some issue that may require a mere warning) is detected during the execution of a job, the job will be aborted, i.e., control will be transferred to the SYSTEM which is responsible for issuing an appropriate error message, destroying the process, releasing the allocated memory, updating the relevant tables, spooling out the job, and releasing the allocated DISK space. The output spooling information will be essentially the same as for a normal termination as much as possible.

5) If a process references a page currently not in the main memory, a page fault will occur and control will be transferred to the Page Fault Handler via the SYSTEM. The process which generated the page fault will be placed on the blocked queue, on which it will remain for 20 virtual time units.

6) If a job references the Input-Data Segment or the Output-Data Segment for the first time, a segment fault will occur and control will be transferred to the Segment Fault Handler via the SYSTEM. Subsequently, a page has to be transferred from the DISK to the main memory, therefore the process will be placed on the blocked queue (for 20 vtu because of the required page transfer).

CLOCK INCREMENT

As before, the clock is associated with the CPU, i.e., it will only be incremented during CPU cycles. Note that the run time for each job consists of CPU time (or pure execution time), time spent for page transfer, etc. Turnaround time may include waiting time.

METERING AND REPORTING FACILITY

You are to introduce the appropriate data structures, probe points, and reporting facilities so that your operating system will provide the following information in addition to the existing job-level output of Phases I and II.

o The number of jobs processed during a simulation run.

o The maximum, minimum, and average CPU time (i.e., user job execution time) used by jobs within the job mix in a simulation run for all the jobs that terminate normally.

o The maximum, minimum, and average user job turnaround time for all the jobs in a simulation run.

o The maximum, minimum, and average code segment size (both as they appear in each batch packet, but as actually computed by the loader).

o The maximum, minimum, and average input segment size (both as they appear in each batch packet, but as the actual number of data items read).

o The maximum, minimum, and average output segment size (both as they appear in each batch packet, but as the actual number of output items written).

o The maximum, minimum, and average number of CPU shots, i.e., the number of times jobs are scheduled to use the CPU.

o The maximum, minimum, and average number of I/O requests for all the jobs in a simulation run.

Note: These performance metrics MUST be collected by the operating system code "on the fly" with the statistics computed at the end of a simulation run. There should be no pre- or post- processing.

Most of the warnings and errors mentioned and discussed in Phases I and II still apply. In addition, as the jobs are processes one by one, the job stream processor must perform error checking to detect errors in the job control records as outlined in Phase II. Some additional examples of possible errors in job stream processing and spooling include extra unused data, attempt to read beyond the end of the input segment, attempt to write beyond the end of the output segment, and program too long to fit on DISK.

Phase III OUTPUT

a) Separate trace files are to be created for the jobs whose trace bits are on and are executing. The names given to such trace files should be a concatenation of the external and internal Ids of each respective job. Note that these files will grow dynamically and incrementally for the active jobs whose trace bits are on.

b) A file called execution_profile is to be created to record all of the significant events that occur system-wide. These events include: job loading time, job transfer time from blocked queue to ready queue, when a warning occurs, when an error occurs, page fault time, job termination time, etc. This file is chronologically ordered based on the virtual time of the occurrence of the events, and must contain at least the following information: the event name, the user job Id, the system assigned job Id, the current clock reading, the information specified in Item 2 on page 1 upon a normal termination, and an error message plus the information specified in Item 4 on page 2 upon an abnormal termination.

The execution_profile must also include the status of the operating system at regular time intervals. The interval size can change depending on the intended usage (e.g., debugging or just observing the behavior of the system). The interval size is to be determined by you, and it should not be too small so that an unnecessarily large file is not generated. The status information should contain the contents of the ready queue, the Id of the job currently executing, the contents of the blocked queue, memory utilization, disk utilization, the current degree of multiprogramming, and anything else that can justifiably enhance the understandability of a timed snapshot of the system.

Also, when the processing of all users' jobs in a simulation run is completed, the system will output the following information at the end of the execution_profile for the entire test batch:

= current value of the clock
= all the info specified in the section on METERING AND REPORTING FACILITY
= the number of jobs that terminated normally
= the number of jobs that terminated abnormally
= total time lost due to abnormally terminated jobs
= total time lost due to suspected infinite jobs
= Id of jobs considered infinite
= mean turnaround time of jobs that terminated normally
= mean waiting time of jobs that terminated normally
= mean number of page faults
= etc.

When reporting averages and utilization numbers, pay special attention to the number of meaningful significant digits.

Note that other requirements, including the write-up on software engineering issues, are as mentioned in Phases I and II.

Two test batches will be made available shortly for running your Phase III operating system: one correct test batch and one test batch with injected errors in order to exercise your error_handler. In the mean time, you can use the test job provided and your own test job, as well as some variants of those jobs with injected errors, to create your own batch of test jobs.

As usual, you are to keep up with class discussions about the programming project. Some specifics of the project may change slightly as a result of the discussions in class.

The naming standards, documentation guidelines, and the late penalty policy as stated in the specifications for Phases I and II apply to Phase III as well.

Summary of the deliverables for Phase III (for both test batches) in addition to your Phase III listing (modular, readable, and well-documented):

- A print-out of the execution_profile (not the entire file, but as specified in the "procedure") for both the correct batch and the batch with injected errors.
- A selection of some of the trace files (sampled as specified in the "procedure").
- The Software Engineering Issues report for Phase III.
- Your marked and graded Phase II project deliverables.

Additional Note:

The incoming jobs (i.e., the "input" to your operating system) represents the stream of jobs being submitted to your operating system. It may seem conceptually/functionally the same whether you read the incoming jobs from a file or from a separate data structure. However, there are several issues (four points are mentioned below) that invalidate the assumption.

- There is no "separate data structure" in a computer system unless we assume the existence of a second disk, but then that disk would need to be "managed" as well.

- The job at the front of the list is arriving "now" and the rest of the jobs will actually be arriving "in the future", so to speak. So a job is outside the system until it is loaded. The "load time" impacts the information collected and reported for each job.

- At the demonstration time, you will be asked to run your operating system on a set of jobs (an input "file") which may be of any size (unpredictable), so the assumption that the input file will always fit in a presumed "data structure" fails.

- Above all, the working of your operating system must not depend on how many jobs there are in the input file. It should work for a set of jobs of size one as well as for a set of jobs of size 200.

Trashing - Gnatt chart