OS-344

Assignment-2

Doubt Clearing session

Presented by

Amit Puri

Teaching Assistant

Focus On

1. Process related system calls
2. Adding user defined information in xv6 Process control block
3. Adding estimated burst-times to process control block
4. Adding new scheduler
5. Adding hybrid scheduler
6. Testing Programs

Part-A

1)

GetNumProc()

Print total number of all types of processes

getMaxPid()

Print the ID of process with maximum process ID in the table.

- Fixed size Process table in xv6, once the OS starts running.
- Process states: Embryo, Running, Runnable, Sleeping, Zombie and Unused.

Part-A

2)

getProcInfo(pid, &processinfo)

It should return:

Parent Process ID, Number of context switches, Process Size.

You may want to print other information as well like Process ID, Process Name and Process Burst Time(To be added) which may be useful in Part-B.

Struct processinfo → (User-defined structure in **processInfo.h)**

Patch file that is provided needs to be added in xv6 directory, to be able to use this structure.

You should also link the header file to required files, to be able to use it.

Part-A

3)

set_burst_time(int n)

get_burst_time()

Where burst of time can be assumed in between 1 to 20 units for any process, which is an estimated amount of time for any user program.

Till this point, only burst time information needed to be added and retrieved.

Part-B

1) Adding a Shortest Job First scheduler in xv6:

What things to take care of?

1. Burst time is just an estimated time for the process and is inputted by user. It is not a real measure of how much time will be actually taken by that process.

2. The function where scheduling algorithm is implemented .

3. Initial burst_time of all the process, when processes are created in system (You may want to set this less than all of your processes, to allow system processes to run first) .

Part-B

2) Adding a Hybrid scheduler in xv6:

(SJF + Round Robin) → Sort the processes according to time quanta in process table, set the time quanta to process with lowest burst time.

What things to take care of?

1. In the scheduling loop, you don’t need to try to implement the process for the actual burst time that is set using set_burst_time (There is no real concept of time-quantum in xv6).

2. Instead, try to manipulate the number of chances given to process with shortest burst time and give equal chances from there on to all processes using round-robin.

Part-B

Testing

- Create user program where you can test you new scheduling algorithm.
- Fork multiple child processes with different burst times and print the order in which processes finishes their execution (Read process management in xv6 and see process system_calls available in xv6 code).
- Add dummy delays in your code to simulate the burst_time of the process.
- Your program might not show the desired results of right execution order.
- Multiple CPUs are simulated, so if one CPU is running a process, other will not take that into account while finding a new one with shortest burst.
- Simulate only 1 CPU.

Submission

- Add user program for each system call that you added in Part-A.
- Name the user program as the name of system call.
- Add a user program to test your scheduler, while using suitable system_calls required for it.
- Collect all the user programs you made for this assignment and only those code files where you added you system calls and new scheduler.
- Create a patch file of your code including makefile and test programs.
- Make a zip file and submit as GroupNumber.zip

Questions?

Thanks....