REPORT – CSE Assignment 5

Name : Dolton M. Fernandes
Roll No : 2018111007

1) waitx sys-call :

This function is called by the parent process and waits for a child process to exit.

It loops through all the processes and check if the process is in zombie state.

If yes then it deletes that process from the queue and returns the pid of that process an stores the runtime and waittime in the address passed as argument.

Formula Used:

‘p’ is the process

```
rtime = p->rtime;
wtime = p->etime – p->ctime – p->rtime;
```

2) Getpinfo sys-call :

This is the structure passed as argument :

```
struct proc_stat
{

        int pid; // PID of each process
        float runtime; // Use suitable unit of time
        int num_run; // number of time the process is executed
        int current_queue; // current assigned queue
        int ticks[5]; // number of ticks each process has received at each of the 5
                        priority queue

};
```

This function first searches for the process with passed pid in the queue.

Then it copies the respective fields into the proc structure passed.

Scheduling Algorithms :

1) FCFS (First Come First Serve) :

This scheduler goes through all the process in queue and finds the one with least creation time for execution.

If process with pid = 1 (init) or pid = 2 (sh) is encountered , it is executed directly.

Also in FCFS the yield function is blocked which preempts the current running process and goes back to the scheduler.

2) PBS (Priority Based Scheduling) :

set_priority(int priority) function :

Changes the priority of the current running process to the passed value and returns the previous priority.

Algo :

The scheduler goes through all the processes in the queue and finds the minimum priority value (Highest priority).

Then we loop through all the processes in 'RUNNABLE' state and execute only the ones with priority equal to the minimum value we found.

Also check_priority() function is called after every clock cycle which checks if the current running process has the highest priority or not. If yes then it preempts the process by calling yield().

3) MLFQ (Multi Level Feedback Queue) :

In this scheduling algorithm we use FCFS for the first four queues and Round Robin in the last 5th queue.

Extra Declared variables in proc :

int current_queue – Current queue in which the process is.
int ticks[5] – Total ticks recieved at each queue.
int arrival_time – Time (ticks) at which the process entered current_queue
int time_spent – Time spent in current_queue
int cont_time – Continuous running time till now from when context was switched to this process.

We push a forked process to the queue with highest priority (Queue 1).

The scheduler first checks the first queue for runnable processes and run them in a FCFS manner , then the second and so on till the 4th queue.

Processes in the 5th queue are executed in a RR manner.

After this another loop through all the processes check if the wait time in the queue has exceeded the queue wait limit or not. If yes, it pushes the process to the next higher priority queue.

In trap.c where the clock is declared the yield function is also called when the process exceeds the time slice of that queue.

If the process completely used the time slice then it is pushed to the next lower priority queue.

Helper functions used :

void copyitup(struct proc *p) – Pushes the process to next higher queue

void copyitdown(struct proc *p) – Pushes the process to next lower queue

void update_runtime() - This function is called after every clock cycle. It updates the runtime of all running processes.

int get_time(int x) – returns the time slice of the queue number passed as argument.

Question :

If a process voluntarily relinquishes control of the CPU, it leaves the queuing network, and when the process becomes ready again after the I/O, it is inserted at the tail of the same queue, from which it is relinquished earlier (Explain in the report how could this be exploited by a process ?)

Answer :

If suppose a process with high I/O demand comes , then it’ll be stuck in the higher priority queues for a long amount of time and hence it’ll have an advantage over other processes.

Performance Comparison of Different Scheduling Algorithms :

Scheduling Algorithm	**Run Time**	**Wait Time**
RR	44	425
FCFS	16	415
PBS	33	422
MLFQ	34	421

FCFS is taking the least time here because of least switching between processes.