Stephen Gregory
09/29/2020
Exam 1
CS 300

Q1:

after runner: data = {0, -1, -2, -3, -4, -5}

Line A:

Child values are: 0 -1 -2 -3 -4 -5

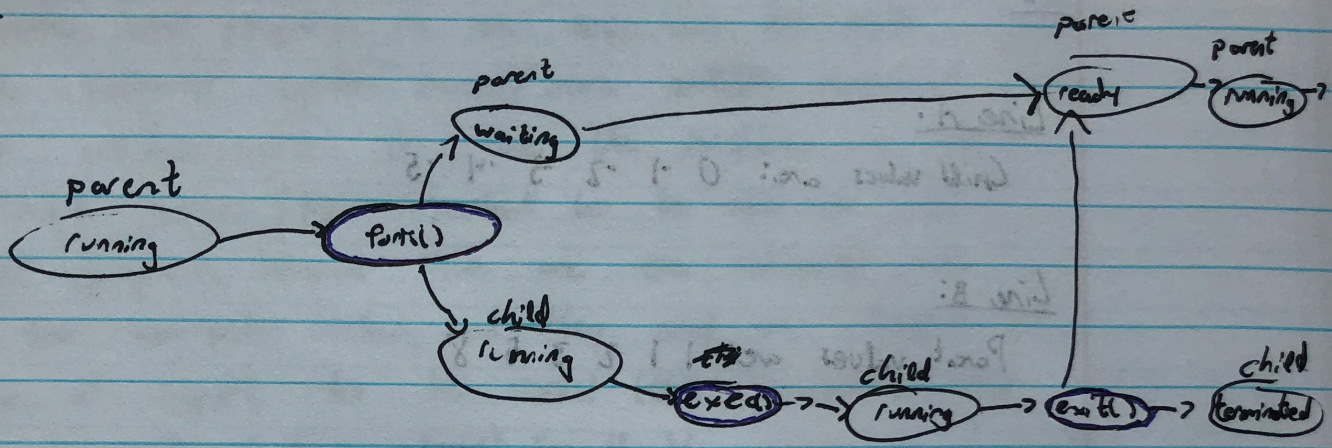
Line B:

Parent values are: 1 1 2 3 5 8

Description:

This program initializes a global array called data, and then uses fork() to create a new child process. In the child process, a new thread is created which executes the function called "runner". This function assigns each value of the global array data to the inverse of it's index. Because threads share global data with their containing process & other threads, this change is seen when the child process prints out the contents of "data". However, the parent process simply prints out the values in it's data global variable, which is obviously different than that of the child process, as processes that are separate don't share global variables.

Q7:

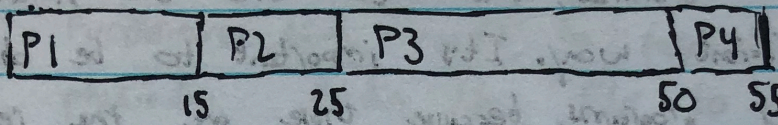


- Blue circle means a function / system call
- no blue circle means process state
- "name" of process shown above process state
- Follows diagram shown in exam
- Note: after child "exit()" call, the parent is in the "ready" state, not "running", because the processor may be doing other tasks in the meantime of the parent's waiting.

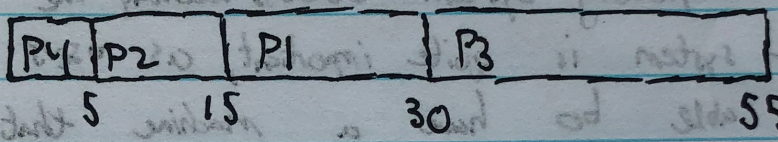
Q10:) The 3 main purposes of an operating system are to execute user programs in an easy way, to make the computer system easier/more convenient to utilize, and to use the hardware in an efficient way. It's important to be able to run user programs, because those are the real-world uses of the computer system, and they need an effective base of support to be useful (that's exactly what the operating system does). Secondly the convenience of the system is quite important, as users want to be able to have a machine that is not only powerful, but convenient & simplistic, as a machine is only as capable as its weakest link, which in many cases of computing, is the human using the system. Lastly, efficiency is of utmost importance, because system resources like CPU executions, memory, etc are all quite limited.

Q18:)

(a) FCFS



SJF



(b)

FCFS	waiting times		SJF	waiting times	
P1	0	units	P1	15	
P2	15	units	P2	5	
P3	25	units	P3	30	
P4	50	units	P4	0	
avg.	(15+25+50)/4 = 22.5	units	avg.	(15+5+30)/4 = 12.5	units

(c)

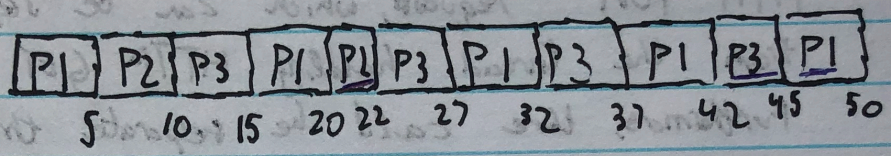
Based on average waiting times alone, SJF seems to be better for the given input (in fact SJF can be shown as provably optimal in theory). The difference in average waiting time is 10 units of time, making SJF considerably better here.

Q19:) A web-server needs to be multithreaded, so that it can handle multiple requests at once. For example, there could be a thread responsible for all HTTP POST requests, which can be separated totally from the thread handling HTTP GET requests. Furthermore, there could be separate threads for physically finding the files to be served to clients of the server out on the internet.

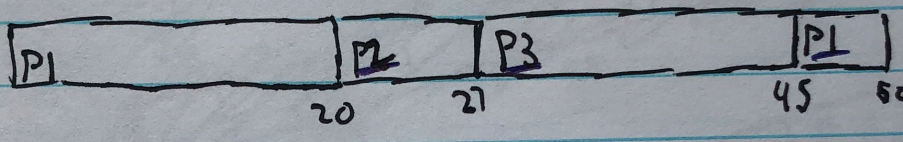
Q20:)

P1 → 25 → 20 → 15 → 10 → 5
P2 → 7 → 2 → 0
P3 → 18 → 13 → 8 → 3

(a) quantum = 5 ms



quantum = 20 ms



P1 → 25 → 5
P2 → 7 → 0
P3 → 18

(b)

quantum(5) response times

P1	0	units
P2	5	units
P3	10	units
avg	$\frac{5+10}{3} = 5$	units

quantum(20) response times

P1	0	units
P2	20	units
P3	27	units
avg	≈ 15.67	units

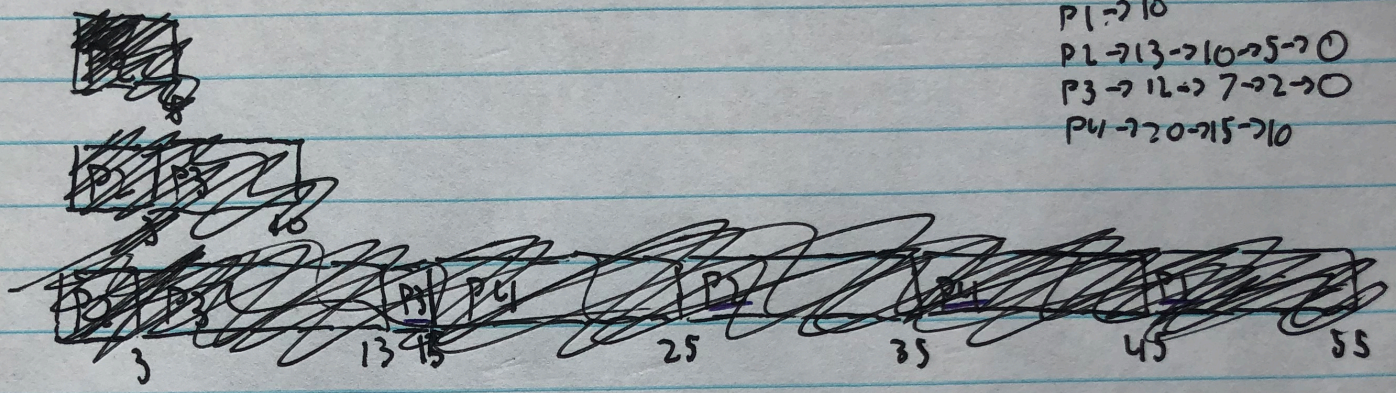
(c) As time quantum increases, our rand-robin algorithm approaches being the same as FCFS. Clearly, for interactive processes, a low time quantum is often better, as response times generally are directly proportional to time quantum (low quantum leads to low response time, & high quantum leads to high response time).

Q21:)

(a) RR quantum = 5

P1 → 10 →
P2 → 13 → 10 → 0
P3 → 12 → 2 → 0
P4 → 20 → 10 → 0

P1 → 10
P2 → 13 → 10 → 5 → 0
P3 → 12 → 7 → 2 → 0
P4 → 20 → 15 → 10



P2	P3	P3	P3	P4	P2	P4	P2	P4	P4	P1	P1

3 8 13 15 20 25 30 35 40 45 50 55

turnaround time		
P1	55-0 = 55	units
P2	35-0 = 35	units
P3	15-3 = 12	units
P4	45-10 = 35	units

response time		
P1	45	units
P2	0	units
P3	3	units
P4	15	units