Assignment -
Processes and CPU scheduling

1. Describe the actions taken by a kernel to context-switch between processes.

Ans Actions taken by a kernel to context - switch between processes are -

- The OS must save the PC and user stack pointer of the currently executing process, in response to a clock interrupt and transfers control to the kernel clock interrupt handler.
- Saving the rest of the registers, as well as other machine state, such as the state of the floating point registers, in the process PCB is done by the clock interrupt handler.
- The scheduler to determine the next process to execute is invoked the OS.
- Then the state of the next process from its PCB is retrieved by OS and restores the registers.

The restore operation takes the processor back to the state in which the previous process was previously interrupted, executing in user code with user-mode privileges.

Many architecture - specific operations, including flushing data and instruction caches also must be performed by Context switches.

2. Construct a process tree similar to Fig 3.8 to

obtain information for the UNIX or LINUX system, use the command ps-ael.

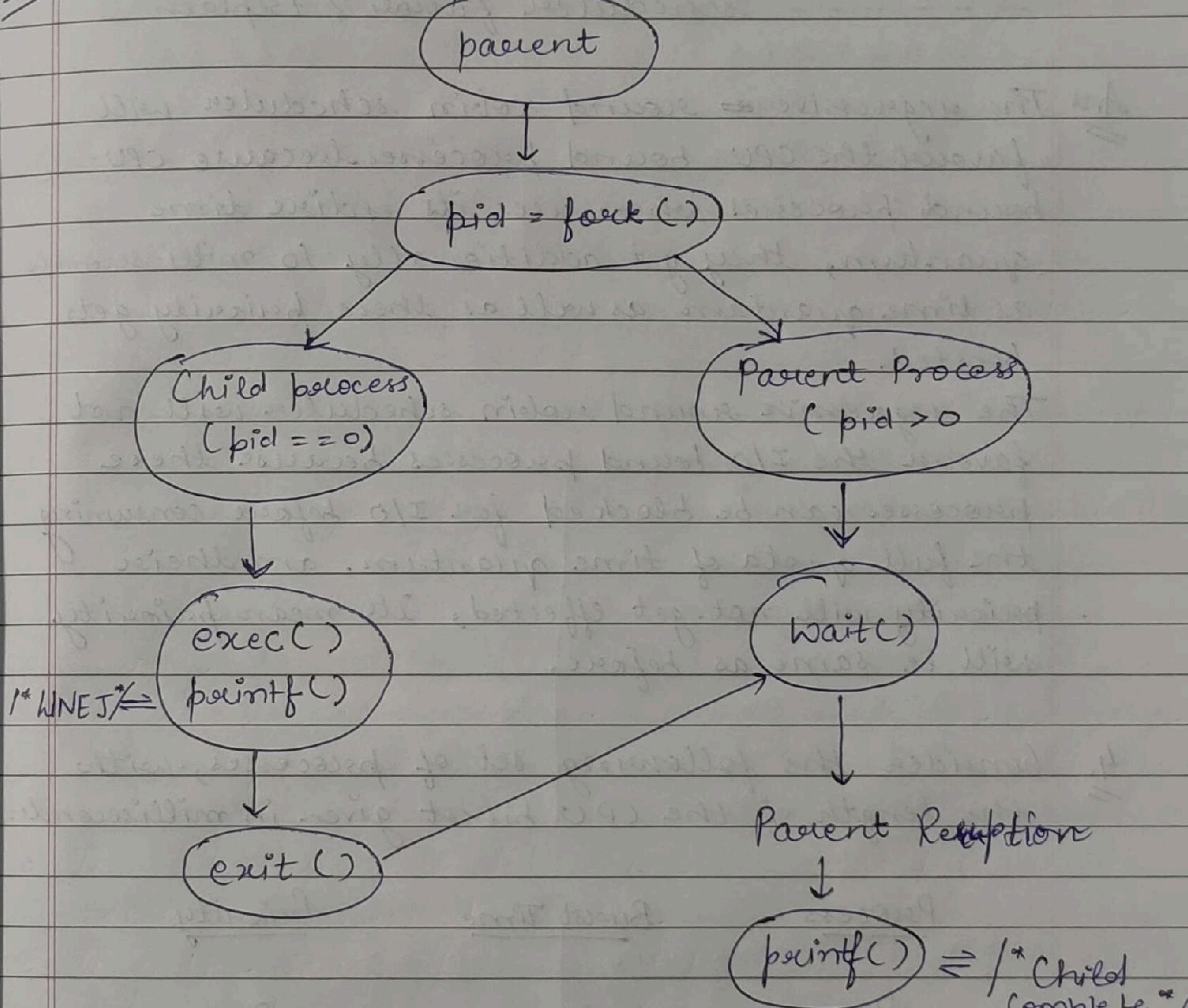
```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>

int main()
{
pid_t pid;

    /* fork a child process */
    pid = fork();

    if (pid < 0) { /* error occurred */
        fprintf (stderr, "Fork Failed");
        return 1;
    }
    else if (pid == 0) { /* child process */
        execlp ("/bin/ls", "ls", NULL);
        printf ("LINE J");
    }
    else { /* parent process */
        /* parent will wait for the child to complete */
        wait (NULL);
        printf ("Child Complete");
    }
    return 0;
}
```

Ans



execlp ( ) list the names of the files and folders existing on the same path (given) as mentioned in the first parameter.

Hence, code followed by first execlp ( ) will never gets executed and the line J will never be reached.

3. A variation of the round-robin ------
------- scheduler favor? Explain

Ans The regressive ~~as~~ round robin scheduler will favour the CPU-bound processes because CPU-bound processes when uses its entire time quantum, they get additionally 10 milli-seconds as time quantum as well as these priority gets boosted.

The regressive round robin scheduler will not favour the I/O bound processes because these processes can be blocked for I/O before consuming the full quota of time quantum, and their priority will not get effected, its mean priority will be same as before.

4. Consider the following set of processes, with the length of the CPU burst given in milliseconds:

Process	Burst Time	Priority
P_1	2	2
P_2	1	1
P_3	8	4
P_4	4	2
P_5	5	3

The processes are assumed to have arrived in the order P_1, P_2, P_3, P_4, P_5, all at time 0.

(a) Draw four Gantt Charts that illustrate the execution of these processes using the following scheduling algorithms : FCFS, SJF, non preemptive priority (a larger priority number implies a higher priority), and RR (quantum = 2).

Ans **FCFS Gantt Chart**

P1	P2	P3	P4	P5

0 2 3 11 15 20

SJF Gantt Chart

P2	P1	P4	P5	P3

0 1 3 7 12 20

Non-Preemptive Priority

P2	P1	P4	P5	P3

0 1 3 7 12 20

RR (Quantum = 2)

P1	P2	P3	P4	P5	P3	P4	P5	P3	P5	P3

0 2 3 5 7 9 11 13 15 17 18 20

(b) What is the turnaround time of each process for each of the scheduling algorithm in part a?

Ans **For FCFS :-**

P_1 : 0 + 2 = 2 ms
P_2 : 2 + 1 = 3 ms
P_3 : 3 + 8 = 11 ms
P_4 : 11 + 4 = 15 ms
P_5 : 15 + 5 = 20 ms

For SJF :-

P_1 : 1 + 2 = 3 ms
P_2 : 0 + 1 = 1 ms
P_3 : 12 + 8 = 20 ms
P_4 : 3 + 4 = 7 ms
P_5 : 7 + 5 = 12 ms.

For Non-Preemptive Priority:-	For RR :-
P_1 : 1 + 2 = 3 ms	P_1 : 0 - 0 + 2 = 2 ms
P_2 : 0 + 1 = 1 ms	P_2 : (2 - 0 - 0) + 1 = 3 ms
P_3 : 12 + 8 = 20 ms	P_3 : (18 - 0 - 6) + 2 = 14 ms
P_4 : 3 + 4 = 7 ms	P_4 : (11 - 0 - 2) + 2 = 11 ms
P_5 : 7 + 5 = 12 ms	P_5 : (17 - 0 - 4) + 1 = 14 ms

(c.) What is the waiting time of each process for each of these scheduling algorithms?

Ans

For FCFS Algorithm-	For SJF Algorithm -
P_1 : 0 - 0 = 0 ms	P_1 : 1 - 0 = 1 ms.
P_2 : 2 - 0 = 2 ms	P_2 : 0 - 0 = 0 ms
P_3 : 3 - 0 = 3 ms	P_3 : 12 - 0 = 12 ms
P_4 : 11 - 0 = 11 ms	P_4 : 3 - 0 = 3 ms
P_5 : 15 - 0 = 15 ms	P_5 : 7 - 0 = 7 ms

For Non-Preemptive Priority	For RR :-
P_1 : 1 - 0 = 1 ms	P_1 : 0 - 0 - 0 = 0 ms
P_2 : 0 - 0 = 0 ms	P_2 : 2 - 0 - 0 = 2 ms
P_3 : 12 - 0 = 12 ms	P_3 : 18 - 0 - 6 = 12 ms
P_4 : 3 - 0 = 3 ms	P_4 : 11 - 0 - 2 = 9 ms
P_5 : 7 - 0 = 7 ms	P_5 : 17 - 0 - 4 = 13 ms

(d) Which of the algorithms results in the minimum average waiting time (over all processes)?

classmate

Date

Page

Ans For FCFS Algorithm,

Average Waiting Time (AWT) $= \frac{0+2+3+11+15}{5} = 6.2$ ms.

For SJF Algorithm,

$AWT = \frac{1+0+12+3+7}{5} = 4.6$ ms

For Non-Preemptive Priority

$AWT = \frac{1+0+12+3+7}{5} = 4.6$ ms

For RR (Quantum = 2)

$AWT = \frac{0+2+12+9+13}{5} = 7.2$ ms.

$\therefore$ SJF Algorithm and Non-preemptive Priority Scheduling results in the minimum average waiting time (AWT).