OPERATING SYSTEMS

Professor: Hungwen Li

HOMEWORK 2

Name: Jeyanthh Venkatachari Ravikumar SJSU ID: 010720528

3.2 8 Processes

3.9 The Actions Taken by the Kernel to Context Switch between Processes are

- When the Context Switching occurs, the CPU must save the state of the current process and restore the state of the new process to be executed form the PCB.
- The context needed to be saved or retrieved during this process includes the Process State, CPU Registers and the memory management information.
- During Context Switching, The kernel saves the context of the old process in the Process Control Block and loads the context of the new process to be executed from the Process Control Block
- Context Switching speed varies depending on the memory speed, the number of registers that must be copied, and the existence of special instructions like load All/Store All

3.12 From the program

```
#include <stdio.h>
#include <unistd.h>
int main() {
int i;
for (i = 0; i < 4; i++){
fork();
}
return 0;
}
```

Each Child Process created, will have the copy of the parent process PCB initially (The value of i will be copied as a local variable)

This value will be incremented and the child will fork again based on the looping conditions.

Therefore total number of processes (including the parent) will be 2^n, where n is the number of times the processes have looped. Here its 4

Therefore 2^4 = 16 Processes.

Name: Jeyanthh Venkatachari Ravikumar SJSU ID: 010720528

3.13 **pid = fork();**

This line will lead to scenarios

1. If the fork is successful, it returns 0, denoting a child process. It will also return the process id of the child process to the parent process.

2. If the forking is unsuccessful, it returns a negative value.

else if (pid == 0) {

/* child process */

execlp("/bin/ls","ls",NULL);

printf("LINE J");}

The above mentioned snippet of code will be executed only when the forking is successful. However, the execlp() command will replace the current process exe and replace it by the exe present tin "/bin/ls"

The "ls" argument present in in execlp()command will terminate the execution of child process.

Therefore the printf(""LINE J") will never be reached in case of proper execution of execlp().
But however, it will execute the printf("LINE J") in case of an exception arising from execution of execlp("/bin/ls","ls",NULL);

3.17

pid = fork();

This line will lead to scenarios

1. If the fork is successful, it returns 0, denoting a child process. It will also return the process id of the child process to the parent process.

2. If the forking is unsuccessful, it returns a negative value.

Considering the fork is successful, it returns 0 and child process ID to the parent, thus executing the 2 IF Statements leading to the following outputs.

Line X: CHILD: 0 CHILD: -1 CHILD: -4 CHILD: -9 CHILD: -16

Line Y: PARENT: 0 PARENT: 1 PARENT: 2 PARENT: 3 PARENT: 4