# Exercises-Chapter 3

# Introduction to Processes

## 1. Using the program shown in Figure below, explain what the output will be at Line A.

## int value = 5;

## int main(){

## pid\_t pid;

## pid = fork();

## if (pid ==0){

## value += 15;

## return 0;

## }

## else if (pid>0) {

## wait(NULL);

## printf(“PARENT, Value = %d”, value); //Line A

## return 0;

}

## }

## 2. Including the initial parent process, how many processes are created by the program shown in Figure below?

## 

## #include <stdio.h>

## #include<unistd.h>

## 

## int main(){

## fork();

## fork();

## fork();

## return;

## }

## 

## 

## 3. When a process creates a new process using the fork() operation, which of the following contents is shared between the parent process and the child process?

#### Stack

#### Heap

#### Shared memory segments created by shmem

4. Define the term context switch. What activities does the OS undertake to context switch from one process to another.

## 5. What information needs to kept in a Process Control Block (PCB) and which ones needs to be saved, changed or updated when context switching takes place.

## 7. Draw the state transition diagram of a process and describe the activities that trigger each transition. Include the states new, ready, running, waiting and terminated. Why is there no transition between ready and waiting or between waiting and running?

## 8. Describe the various process queues in an OS. What event or program cause a PCB move from one queue to the other?

## 

## 9. Describe the differences among short-term, medium-term, and long term scheduling.

## 10. To obtain process information for the UNIX or Linux system in tree format, use the command ps –aef --forest. Name a process that has 2 forked children. How many children does the init process have and describe what each one does?

## 11. Using the program given below, identify the values of pid at lines A, B, C, and D. (Assume that the actual pids of the parent and child are 2600 and 2603, respectively.)

## #include *<*sys/types.h*>*

## #include *<*stdio.h*>*

## #include *<*unistd.h*>*

## int main()

## {

## pid t pid, pid1;

## /\* fork a child process \*/

## pid = fork();

## 

## if (pid < 0) { /\* error occurred \*/

## fprintf(stderr, "Fork Failed");

## return 1;

## }

## else if (pid == 0) { /\* child process \*/

## pid1 = getpid();

## printf("child: pid = %d",pid); /\* A \*/

## printf("child: pid1 = %d",pid1); /\* B \*/

## }

## else { /\* parent process \*/

## pid1 = getpid();

## printf("parent: pid = %d",pid); /\* C \*/

## printf("parent: pid1 = %d",pid1); /\* D \*/

## wait(NULL);

## }

## return 0;

## }