CprE 308 Homework 1

Department of Electrical and Computer Engineering

Iowa State University

Student Name: Khushveen Kaur Umra

University-ID: 947005108

Problem 1 (5 points)

What is a process? What is a "Zombie" process?

- ➔ A process is an instance of a computer program that is currently being executed. It is a running instance of a program. It is made up of the program instruction, data read from files, other programs or input from a system user. There are mainly two types of processes in Linux, i.e., the foreground processes and the background processes. It can either be a parent (creator or other processes during run-time) or child (process created by others) process.
- ➔ A 'Zombie' process is a process whose execution is completed but it still has an entry in the process table. The 'Zombie' process usually occurs for child processes, as the parent process still needs to read its child status. If the parent process calls wait () system call, then the execution of the parent is suspended until the child is terminated. Even though the child is terminated, there is an entry in the process table corresponding to the child where the status is stored. When the parent collects the status, the entry is deleted, leading to removing all the traces of the child process from the system. This state of the child Process is known as the 'Zombie' process.

Problem 2 (5 points)

Compare the advantage & disadvantage of Monolithic kernel and Microkernel.

➔ Monolithic Kernel: (Advantage)
i) Execution of processes is fast due to separate memory space for user and kernels
ii) To write a monolithic kernel, less code is required, which generally means fewer bugs and security problems is also less
iii) It provides CPU scheduling, memory management, file management, and other operating system functions through systems calls.

(Disadvantage) – i) If any one service fails, it will lead to an entire system failure.
ii) In order to add any new service, the user needs to modify the entire operating system.
iii) Coding in the kernel space is very hard and difficult, which can make debugging difficult.

➔ Microkernel: (Advantage)

i) Microkernels has fewer system crashes. It helps in enforcing a more modular system structure.
ii) Microkernels are secure since only those parts are added, which might disturb the system's functionality.
iii) It adds new features without recompiling and the architecture of the microkernel is small and isolated, but it may work better.

(Disadvantages): i) Providing services in a microkernel system are expensive compared to the normal monolithic system

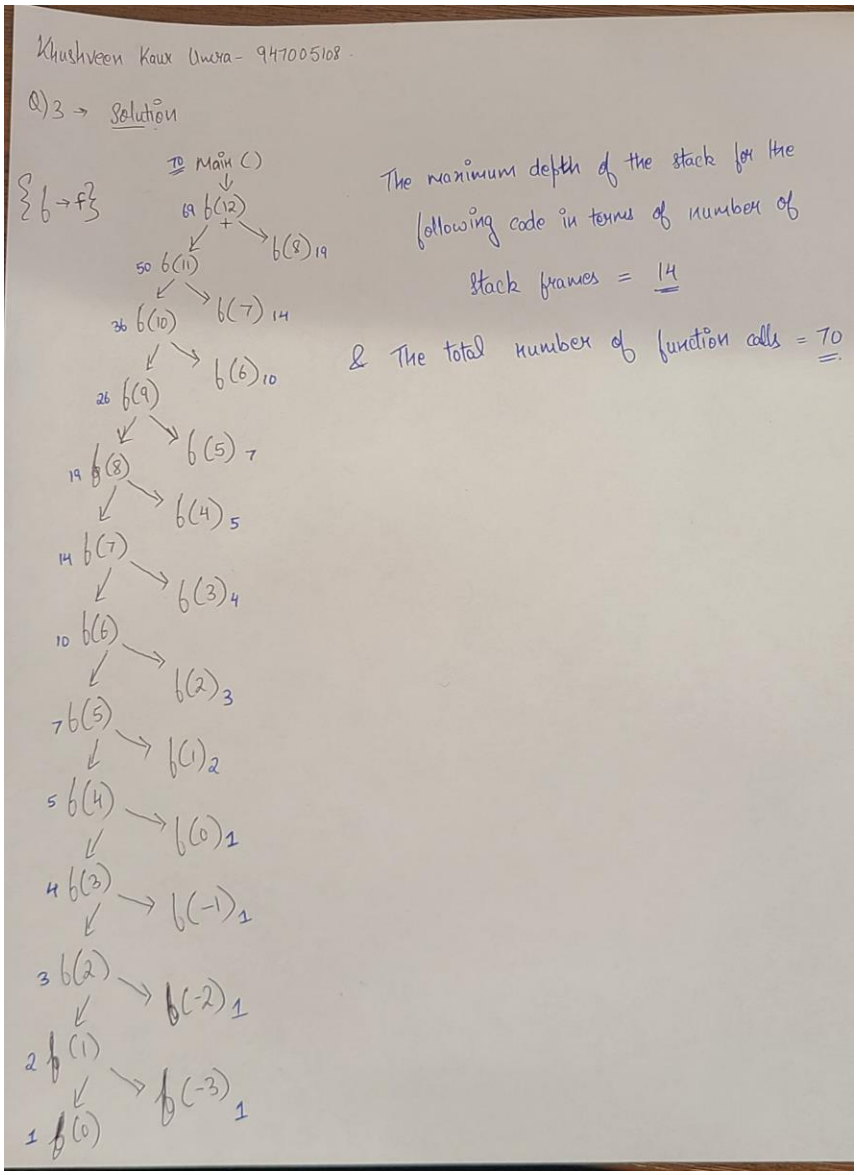
ii) The performance of a microkernel system might be indifferent and cause issues

iii) When the drivers are implemented as procedures, a context switch or a function call is needed

Problem 3 (5 points)

What is the maximum depth of the stack for the following code in terms of number of stack frames (one stack frame per function call)? Be sure to count the first call to main.

```
int main()  {

f(12);

 return 1;  }

int f (int n) {

if (n <= 0)

 return 0;

else return f(n-1) + 2 * f(n-4);  }
```



Problem 4 (5 points)

(1) How many processes does the following code create?

(2) Draw the process tree of the program.

```
 int main() {
int i;
for (i=1; i<3; i++)
fork();
return 1;
}
```

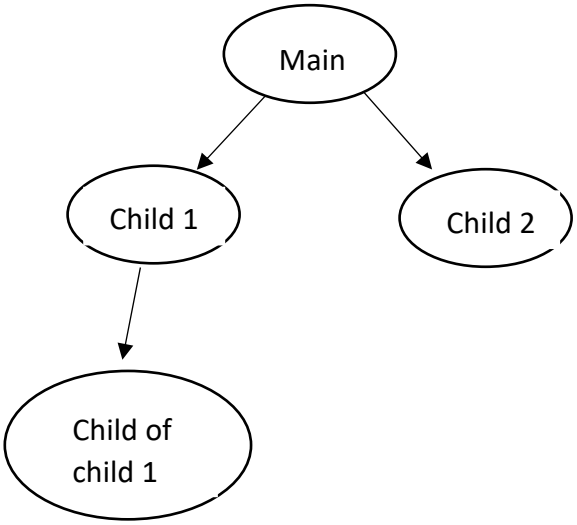
➔ 1) Based on the ‘for’ statement, there will two executions, for i=1 and i=2.

When i=1; the ‘main’ process will create one child process.

When i=2; the ‘main’ process will create one child process. And another process will be created, which will be the child process of the 1st child process.

Hence, the total number of process is equal to ‘4’, in which there are ‘3’ child processes.

2) The process tree:



Problem 5 (5 points)

Please write all possible outputs from the following piece of code:

```
 int main(void) {
pid_t pid = fork();
 if (pid > 0) {
printf("I am the parent\n"); }
else if (pid == 0) {
printf("I am the child\n");
}
else printf("ERROR!\n");
return 0;
 }
```

➔ The possible outputs for the following code are:

i) I am the parent
I am the child

ii) ERROR!

iii) I am the parent

iv) I am the child

Problem 6 (5 points)

Consider the following code. Assume all system calls return successfully and the actual process IDs of the parent and child during the execution are 2600 and 2603, respectively. What are the values of pid/pid1 at lines A, B, C, D?

```
int main() {
        pid_t pid, pid1;
        pid = fork();
if (pid < 0) {
        fprintf (stderr, "Fork Failed");
         return 1;
} else if (pid == 0) {
        pid1 = getpid();
        printf("child: pid = %d", pid); /* A */
         printf("child: pid1 = %d", pid1); /* B */
} else {
        pid1 = getpid();
        printf("parent: pid = %d", pid); /* C */
         printf("parent: pid1 = %d", pid1); /* D */
        wait(NULL);
}
return 0;
}
```

➔ Output:

(Line A) child: pid = 0

(Line B) child: pid1 = 2603

(Line C) parent: pid = 2603

(Line C) parent: pid1= 2600