COMP 3411

Assignment 2

Martin Atanacio - T00684924

C Program

```
[dkcomp341126@cs2 ~]$ vi testOS.c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

int main(int args, char *argv[])
{

    pid_t fork_return;
    pid_t pid;
    pid = getpid();
    fork_return = fork();

    if (fork_return == -1)
    {
        // When fork() returns -1, an error has happened.
        printf("\nError creating process ");
        return 0;
    }

    if (fork_return == 0)
    {
        // When fork() returns 0, we are in the child process.
        printf("\n\nThe values are Child ID = %d, Parent ID=%d\n", getpid(), getppid());
        execl("/bin/cat", "cat", "-b", "-v", "-t", argv[1], NULL);
    }

    else
    {
        // When fork() returns a positive number, we are in the parent process and the return value is
 the PID of the newly created child process.
        wait(NULL);
        printf("\nChild Completes\nIn the Parent Process\n");
        printf("Child Id = %d, Parent ID = %d\n", getpid(), getppid());
    }

    return 0;

}
```

```
[dkcomp341126@cs2 ~]$ ./a.out mytextfile.txt


The values are Child ID = 945, Parent ID=944
     1  This is line one in my text file.
     2  This is line two in my text file.
     3  This is line three in my text file.
     4  Linux provides many commands for manipulating text!

Child Completes
In the Parent Process
Child Id = 944, Parent ID = 819
```

Question 2

1. **If you try to print a message after the *exec** call, does it print it? Why? Why not?**
   No, the exec* call replaces the file with the new one (the one that is specified in exec*), hence it will not execute past that line.

2. **Who is the parent of your executable program?**
   The shell (in this case, bash).

3. **How would you change the code so that the child and parent run concurrently?**
   Remove the wait(NULL) function call. The scheduler would then have to determine which one executes first.

Question 3

1. **Run a process in background**
   Append an '&' at the end of the command. Output will show the job number alongside its PID.

```
[dkcomp341126@cs2 ~]$ cat mytextfile.txt &
[1] 1555
[dkcomp341126@cs2 ~]$ This is line one in my text file.
This is line two in my text file.
This is line three in my text file.
Linux provides many commands for manipulating text!

[1]+  Done                    cat mytextfile.txt
```

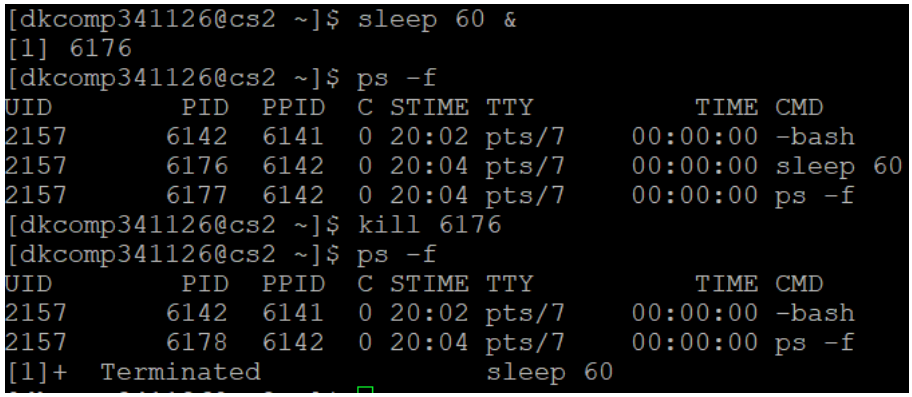
2. **Run a process in foreground**
   Run the process/command without appending the '&' at the end. Output will show on screen.

```
[dkcomp341126@cs2 ~]$ cat mytextfile.txt
This is line one in my text file.
This is line two in my text file.
This is line three in my text file.
Linux provides many commands for manipulating text!
```

3. **Call the process from background to foreground**
   Use the 'fg' command. Find the job number using the 'jobs' command. Command will bring the process to screen.

```
[dkcomp341126@cs2 ~]$ sleep 60 &
[1] 2297
[dkcomp341126@cs2 ~]$ sleep 60 &
[2] 2300
[dkcomp341126@cs2 ~]$ sleep 60 &
[3] 2301
[dkcomp341126@cs2 ~]$ fg 3
sleep 60
```

4. **Kill the process**
   Use the ‘kill [pid]’ command (with the process ID - found by using the ‘ps’ command). Process is killed and shown in the job list as ‘Terminated’.



5. **Show the process status**
   Use the ‘ps -f’ command to show a full list of process’ status. Output will show a list of processes with their corresponding PID, PPID, etc.