CS342 Operating Systems

Homework 2 Report

Ege Türker 21702993 Section 1

Instructor

İbrahim Körpeoğlu

Spring 2021

Q1)

```c
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/wait.h>


int createProcesses(int k){

   printf("Root process id: %d \n", getpid());

   int left, right;
   for( int i = 1; i < k; i++){

       left = fork();

       if( left != 0)
           right = fork();
       if( right != 0 && left != 0)
           break;
       printf( "Child process id: %d Parent process id: %d
\n",getpid(), getppid());
   }

   int* status;
   waitpid(left, status, 0);
   waitpid(right, status, 0);

   return(0);

}

int main(){
   int k;

   do{
       printf("Enter k value between 1 and 5:  \n");
       scanf("%d", &k);
   }while (k > 5 || k < 0);

   createProcesses(k);
}
```

Q2)

```
long counter;
long nice;
unsigned long policy;
struct mm_struct *mm;
int processor;
struct list_head run_list;
unsigned long sleep_time;
struct task_struct *next_task, *prev_task;
struct mm_struct *active_mm;
struct list_head local_pages;
```

Q3)

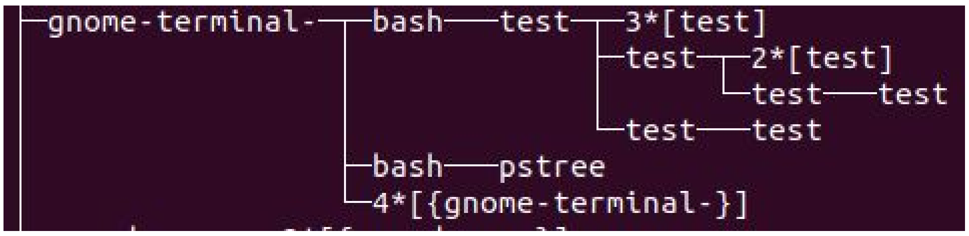
I wrote this code to see how many processes were created:

```
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/wait.h>

int main(){
   fflush(stdout);
   pid_t n;
   for(int i = 0; i < 5; ++i){
       n = fork();
       fflush(stdout);

       if(n == 0){
           printf("Child %d - Parent %d \n", getpid(), getppid());
           if(i%2 == 0){
               printf("Exit %d \n", getpid());
               fflush(stdout);
               exit(0);
           }
       }
   }

   sleep(10000);
}
```

The program printed this output:

```
Child 3691 - Parent 3686
Exit 3691
Child 3690 - Parent 3686
Child 3689 - Parent 3686
Exit 3689
Child 3688 - Parent 3686
Child 3687 - Parent 3686
Exit 3687
Child 3692 - Parent 3690
Exit 3692
Child 3695 - Parent 3688
Exit 3695
Child 3694 - Parent 3688
Child 3693 - Parent 3688
Exit 3693
Child 3696 - Parent 3694
Exit 3696
```

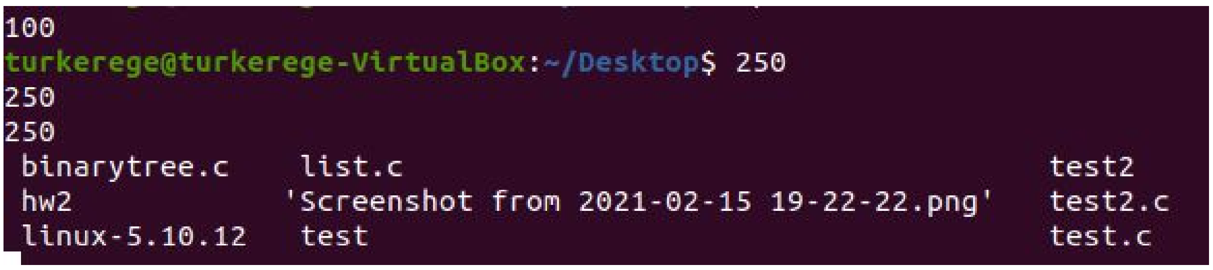
3686 is the main process. So 10 other processes were created with fork().
pstree also confirms this. There are 10 test processes under the first parent process.



```
 ├─gnome-terminal-─┬─bash───test─┬─3*[test]
 │                 │             ├─test─┬─2*[test]
 │                 │             │      └─test───test
 │                 │             └─test───test
 │                 ├─bash───pstree
 │                 └─4*[{gnome-terminal-}]
```

Q4)

100 is printed once, 250 is printed three times.
ls is called once to list the current directory.



```
100
turkerege@turkerege-VirtualBox:~/Desktop$ 250
250
250
 binarytree.c    list.c                                       test2
 hw2            'Screenshot from 2021-02-15 19-22-22.png'     test2.c
 linux-5.10.12   test                                         test.c
```

Q5)

```
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/wait.h>

int main(){

   pid_t child1, child2;

   child1 = fork();

   if (child1 == 0){
       execlp("/bin/ps", "ps","aux", NULL);
   } else{
       child2 = fork();
       if (child2 == 0){
           execlp("/bin/ls", "ls", "-al", NULL);
       }
       else{
           int status;
           waitpid(child1, &status, 0);
           waitpid(child2, &status, 0);
           return(0);
       }
   }
}
```

Q6)

```c
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/wait.h>
#include <string.h>

int main(){

   char write_msg[65] = "I hear and I forget. I see and I remember. I do and I understand.";
   char read_msg[65];

   int fd[2];
   pid_t child1, child2;

   pipe(fd);

   child1 = fork();

   if (child1 == 0){
       close(fd[0]);
       write(fd[1], write_msg, 66);
       close(fd[1]);
   } else{
       child2 = fork();
       if (child2 == 0){
           close(fd[1]);
           read(fd[0],read_msg,65);
           printf("Child2 reads: %s \n", read_msg);
           close(fd[0]);
       }
       else{
           int status;
           waitpid(child1, &status, 0);
           waitpid(child2, &status, 0);
           return(0);
       }
   }
}
```

Q7)

```c
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/stat.h>
#include <fcntl.h>

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

   int from, to, bitCount;
   char buffer;

   if(argc != 3){
       printf("Run with arguments <fileNameToCopyFrom>
<fileNameToCopyTo> \n");
       return -1;
   }

   from = open(argv[1], S_IRUSR);
   to = creat(argv[2], S_IRWXU);

   bitCount = lseek(from, (off_t)0, SEEK_END);

   for( int i = 0; i < bitCount -1; i++){
       lseek(from, (off_t)i, SEEK_SET);

       read(from, &buffer, 1);

       write(to, &buffer, 1);
       write(to, &buffer, 1);
   }

   close(from);
   close(to);
   printf("Done. \n");
}
```