SCR2043 OPERATING SYSTEMS
Lab 2

Name : Nurul Syamira binti Amat Jifri

Marks

Student ID : A19EC0145 Section : 07

Example: An execution of program 1 produces a process tree, an output, and process identifier assignment as shown in Figure 1. The parent process identifier is assumed to have been assigned with an integer 21, while the child is assigned with the next available identifier i.e 22.

```
# include <stdio.h>
# include <unistd.h>
main() {
       int pid1;
       printf(" \n Hello World\n");
       pid1=fork();
       if (pid1 < 0 )
              printf("\n Error in forking \n")
       else if (pid1 == 0)
              printf("\n Knock! Knock! Knock!\n");
       else (pid1 > 0)printf("\n Please come in \n");
}
```

program 1

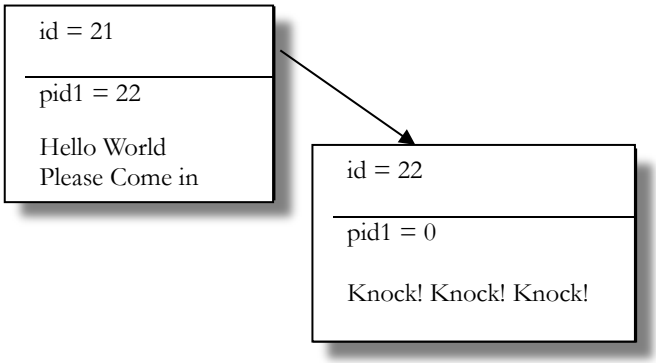


Figure 1: Process tree of program 1

Q1 : Firstly, understand the coding and rewrite back using UNIX program. Then, draw a process tree (as in Figure 1) to illustrate parent-child relationship for the C code given in program 2. For each node on the tree, write the following information:

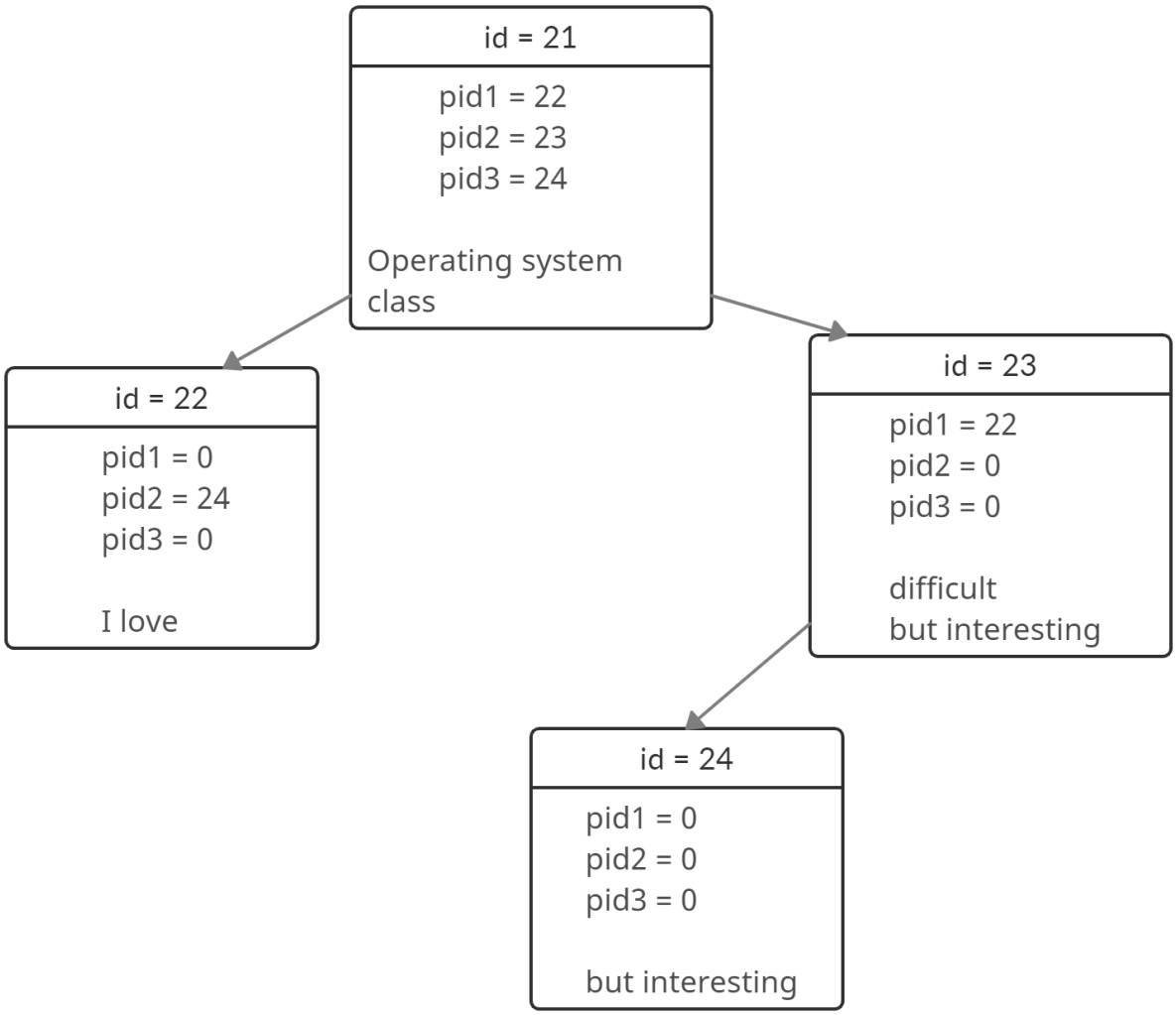
i. The process identifier (assumed that the parent process identifier is assigned with , **id=21**).
ii. The return value of the `fork()` statement
iii. Generated output

```
# include <stdio.h>
# include <unistd.h>

main()
{
        int pid1, pid2, pid3;
        pid1=fork();
        if (pid1 == 0)
        printf("\n I love \n");
        else
        {
        printf("\n Operating system\n");
        pid2=fork();
        if (pid2 == 0)
        printf("\n class, \n");
        else
        {
        printf("\n difficult \n");
        pid3=fork();
        printf("\n but interesting!!\n");
        }
        }
}
```

Program 2: Generation of child process

Answer Q1:


id = 21
pid1 = 22
pid2 = 23
pid3 = 24
Operating system
class
id = 22
pid1 = 0
pid2 = 24
pid3 = 0
I love
id = 23
pid1 = 22
pid2 = 0
pid3 = 0
difficult
but interesting
id = 24
pid1 = 0
pid2 = 0
pid3 = 0
but interesting

Q2: Referring to Figure 2, write a full program that will create five processes called original, clone1, clone2, clone3, and clone4. Each process displays message that illustrate the parent - child/children relationship between two processes.

(8 points)

i. Process *original* will print an output as follows: (*2.5 points)*

" I am the original process, my process identifier is *id_of_original*, my clones are: "
process clone1 : *id_of_clone1*
process clone3 : *id_of_clone3*"

ii. Process *clone1* will print an output as follow: (*1.5 points)*

"I am the obsolete version, recent version is *id_of_clone2"*

iii. Process *clone4* will print the following message: (*1.5 points)*

" I am the most sophisticated clone, process identifier of my parent is *id_of_clone3 "*

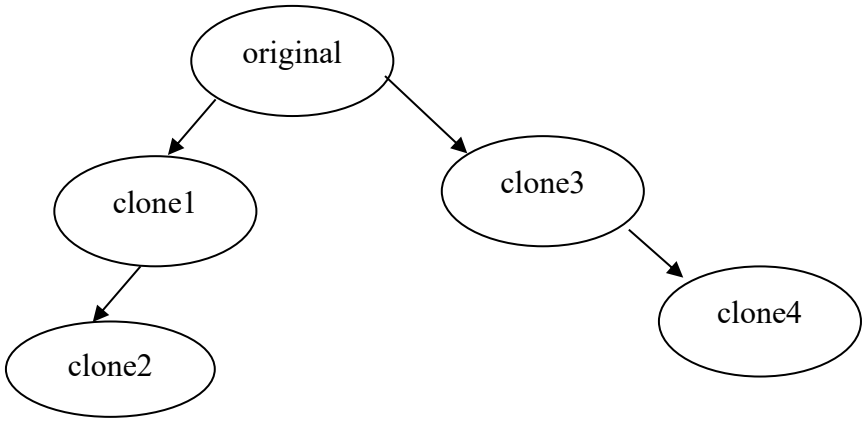


Figure 2: Tree of five processes

Answer Q2:

```
#include <stdio.h>
void original()
{
     printf("I am the original process, my process identifier is
     %d my clones are \n", 0);
     clone1();
     clone3();
}

void clone1()
{
     printf("I am obsolete version, recent version is %d\n", 1);
     clone2();
}

void clone2()
{
     printf("I clone of the %d\n", 2);
}

void clone3()
{
     clone4();
}

void clone4()
{
     printf("I am the most sophisticated clone, process identifier
     of my parent is %d\n", 3);
}

int main()
{
     original();
}
```