Exercise Series N°3.1 : Unix / Linux Processes

Exercise IV.1: (Understanding Questions)

Q1) What is the difference between the **system** and **exec** functions?

Q2) In Unix, Which system call creates the new process?

Q3) What is a zombie process?

Q4) Why all processes are technically zombie processes for at least a brief time. (Hint: think about which of the five process states zombie processes must exist in.)

Q5) What is an orphan process?

Q6) Why UNIX systems reassign processes to a new "parent" rather than simply terminating a process as soon as it's orphaned.

Exercise IV.2: (Process Creation with the fork System Call)

Consider the following program.

```
1 : int main() {
2 :    pid_t pid;
3 :    int x = 1;
4 :    pid = fork();
5 :    if (pid == 0) {
6 :         printf("In child: x=%d\n", ++x);
7 :         exit(0);
8 :    }
9 :    printf("In parent: x=%d\n", --x);
10 :   exit(0);
11 : }
```

Q1) Assuming all **fork** invocations are successful, what does the following program display when run on the Unix/Linux OS?

Q2) How many "**hello**!" lines does each of the following programs print?

Program 1

```
1 : int main() {
2 :    int i;
3 :    for (i=0; i<2; i++)
4 :         fork();
5 :    printf("hello!\n");
6 :    exit(0);
7 : }
```

Program 2

```
1 : void doit() {
2 :    fork();
3 :    fork();
4 :    printf("hello!\n");
5 : }
6 : int main() {
7 :    doit();
8 :    printf("hello!\n");
9 :    exit(0);
10 : }
```

Program 3

```
1 : int main() {
2 :    if (fork())
3 :       fork();
4 :    printf("hello!\n");
5 :    exit(0);
6 : }
```

Program 4

```
1 : int main() {
2 :    if (fork()==0)
3 :       if (fork()) {
4 :         printf("hello!\n");
5 :       }
6 :    exit(0);
7 : }
```

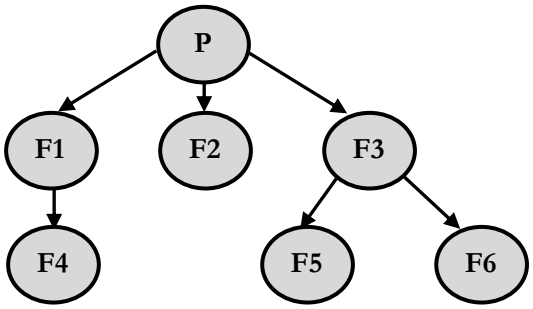
Exercise IV.3: (Hierarchical Process Tree)

Draw the hierarchical process tree created by running the following program.

```
1 : #include <unistd.h>
2 : int main()
3 : {
4 :    fork() && (fork() || fork());
5 :    return 0;
6 : }
```

Exercise IV.4: (Writing a Program)

Write a program to generate the family (genealogical) tree shown in the fallowing figure.



Exercise IV.5: (Number of Process Created & Family Tree)

Indicate the number of processes created and draw the family tree of processes generated by the following programs:

Program 1	Program 2	Program 3

1: **int** main()
2: {
3: fork();
4: fork();
5: fork();
6: }
	1: **int** main()
	2: {
	3: **if** (fork() > 0)
	4: fork();
	5: }
		1: **int** main() {
		2: **int** cpt=0;
		3: **while** (cpt < 3) {
		4: **if** (fork() > 0)
		5: count++;
		6: **else**
		7: count=3;
		8: }

Exercise IV.6: (Family Tree)

Consider the following C program:

```
1: int main (){
2:    pid_t p1, p2, p3, p4;
3:    if ((p1 = fork ()) == 0)
4:       if ((p2 = fork ()) == 0)
5:          f2 ();
6:       else
7:          f1 ();
8:    else
9:       if ((p3 = fork ()) == 0)
10:         f3 ();
11:      else
12:         if ((p4 = fork ()) == 0)
13:            f4 ();
14:   sleep (3);
15:   while (wait (NULL) > 0);
16: }
```

Draw the process tree created by this program if functions **f1**, **f2**, **f3** and **f4** end with **exit()**.