22AIE202 – OPERATING SYSTEMS

LABSHEET 4

Name : Anuvind MP

Roll no: AM.EN.U4AIE22010

---

1. Execute the above program more than once. What is the order in which the processes are being executed? Is it the same in every execution?

CODE :

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

int main(){
    pid_t pid = getpid();
    pid_t ppid = getppid();
    printf("Label -> A PID -> %d PPID -> %d\n", getpid(), getppid());
    if(fork()){
        wait(NULL);
        if(fork()){
            wait(NULL);
            if(!fork()){
                printf("label -> D PID ->  %d PPID -> %d\n", getpid(), getppid());
                if(!fork()){
                    printf("label -> G PID -> %d PPID -> %d\n", getpid(), getppid());
                    if(fork()){
                        wait(NULL);
                        if(!fork()){printf("label -> I PID -> %d PPID -> %d\n", getpid(),
getppid());}
                        else{wait(NULL);}}
                    else { printf("label -> H PID -> %d PPID -> %d\n", getpid(),
getppid()); }}
                else { wait(NULL); }}
            else { wait(NULL); }}
        else{
            printf("label -> C PID -> %d PPID -> %d\n", getpid(), getppid());
            if(fork()){
                wait(NULL);
                if(!fork()) { printf("label -> F PID -> %d PPID -> %d\n", getpid(),
getppid()); }
                else { wait(NULL); }}
            else { printf("label -> E PID -> %d PPID -> %d\n", getpid(), getppid()); }}}
    else{printf("label -> B PID -> %d PPID -> %d\n", getpid(), getppid());}
    return 0;}
```

OUTPUT :

```
root@anuvind:~/LABSHEET4# ./lab4q1
Label -> A PID -> 469 PPID -> 384
Label -> B PID -> 470 PPID -> 469
Label -> C PID -> 471 PPID -> 469
Label -> E PID -> 472 PPID -> 471
Label -> F PID -> 473 PPID -> 471
Label -> D PID -> 474 PPID -> 469
Label -> G PID -> 475 PPID -> 474
Label -> H PID -> 476 PPID -> 475
Label -> I PID -> 477 PPID -> 475
```

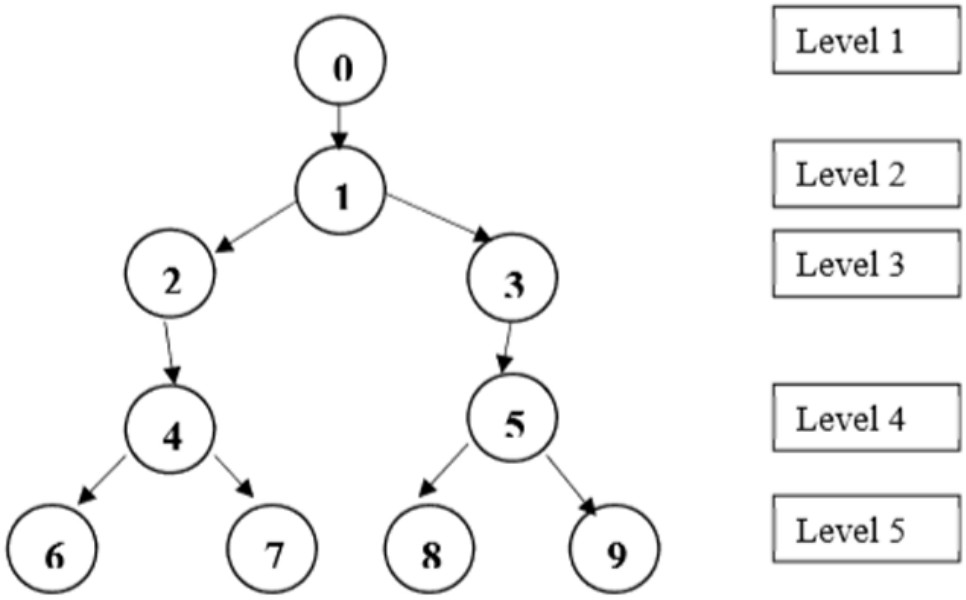
```
root@anuvind:~/LABSHEET4# ./lab4q1
Label -> A PID -> 430 PPID -> 367
Label -> B PID -> 431 PPID -> 430
Label -> C PID -> 432 PPID -> 430
Label -> E PID -> 433 PPID -> 432
Label -> F PID -> 434 PPID -> 432
Label -> D PID -> 435 PPID -> 430
Label -> G PID -> 436 PPID -> 435
Label -> H PID -> 437 PPID -> 436
Label -> I PID -> 438 PPID -> 436
```

```
root@anuvind:~/LABSHEET4# ./lab4q1
Label -> A PID -> 443 PPID -> 367
Label -> B PID -> 444 PPID -> 443
Label -> C PID -> 445 PPID -> 443
Label -> E PID -> 446 PPID -> 445
Label -> F PID -> 447 PPID -> 445
Label -> D PID -> 448 PPID -> 443
Label -> G PID -> 449 PPID -> 448
Label -> H PID -> 450 PPID -> 449
Label -> I PID -> 451 PPID -> 449
```

INFERENCE :

The sequence of process execution remains consistent in the provided code, yet each time it runs, the process IDs vary. This means that while the order of operations remains unchanged, the specific identification numbers assigned to each process differ with each execution.

2. Write a program to create processes according to the tree structure given below. All processes should print their Process id and Parent Process id and the label given in the diagram.



CODE :

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

int main() {
    pid_t pid = getpid();
    pid_t ppid = getppid();
    printf("Label -> 0 PID -> %d PPID -> %d\n", getpid(), getppid());
    if(!fork()){ // process 1
        printf("Label -> 1 PID -> %d PPID -> %d\n", getpid(), getppid());

        if(!fork()){ // process 2
            printf("Label -> 2 PID -> %d PPID -> %d\n", getpid(), getppid());

            if(!fork()){ // process 4
                printf("Label -> 4 PID -> %d PPID -> %d\n", getpid(), getppid());

                if(!fork()){ // process 6
                    printf("Label -> 6 PID -> %d PPID -> %d\n", getpid(), getppid());}

                else{ // process 4
                    wait(NULL);

                    if(!fork()){ // process 7
                        printf("Label -> 7 PID -> %d PPID -> %d\n", getpid(), getppid());}
                    else{wait(NULL);}}}
            else{wait(NULL);} // process 2
        }
```

```
        else{ // process 1
            wait(NULL);
            if(!fork()){ // process 3
                printf("Label -> 3 PID -> %d PPID -> %d\n", getpid(), getppid());

                if(!fork()){ // process 5
                    printf("Label -> 5 PID -> %d PPID -> %d\n", getpid(), getppid());

                    if(!fork()){ // process 8
                        printf("Label -> 8 PID -> %d PPID -> %d\n", getpid(), getppid());}
                    else{ // process 5
                        wait(NULL);
                        if(!fork()){printf("Label -> 9 PID -> %d PPID -> %d\n", getpid(),
getppid());} // process 9
                        else {wait(NULL);}}} // process 8
                else{wait(NULL);} // process 3
            }
            else{wait(NULL);}}  // process 1
    }
    else {wait(NULL);} // process 0

    return 0;
}
```

OUTPUT :

```
root@anuvind:~/LABSHEET4# gcc lab4q2.c -o lab4q2
root@anuvind:~/LABSHEET4# ./lab4q2
Label -> 0 PID -> 507 PPID -> 384
Label -> 1 PID -> 508 PPID -> 507
Label -> 2 PID -> 509 PPID -> 508
Label -> 4 PID -> 510 PPID -> 509
Label -> 6 PID -> 511 PPID -> 510
Label -> 7 PID -> 512 PPID -> 510
Label -> 3 PID -> 513 PPID -> 508
Label -> 5 PID -> 514 PPID -> 513
Label -> 8 PID -> 515 PPID -> 514
Label -> 9 PID -> 516 PPID -> 514
```

3. Write a program to find the area and perimeter of circle and square. Create separate processes to perform the calculation of circle and square.

CODE:

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

```c
int main() {
    int r;
    float Carea, Cperimeter;

    printf("Enter the radius of the circle: ");
    scanf("%d" , &r);

    int a;
    printf("Enter the side of the square: ");
    scanf("%d" ,&a);

    if(fork()){
        printf("Label -> CIRCLE PID -> %d PPID -> %d\n" , getpid() , getppid());
        Carea = 3.14 * r * r;
        Cperimeter = 2 * 3.14 * r;
        printf("The area of circle is %f and perimeter is %f\n", Carea, Cperimeter);
        }
    else{
        printf("Label -> SQUARE PID -> %d PPID -> %d\n", getpid(), getppid());
        printf("Area of square is %d and perimeter is %d\n", (a*a), (4 * a));
    }
    return 0;}
```

OUTPUT:

```
root@anuvind:~/LABSHEET4# gcc lab4q3.c -o lab4q3
root@anuvind:~/LABSHEET4# ./lab4q3
Enter the radius of the circle: 7
Enter the side of the square: 5
Label -> CIRCLE PID -> 615 PPID -> 384
Label -> SQUARE PID -> 616 PPID -> 615
Area of square is 25 and perimeter is 20
The area of circle is 153.860001 and perimeter is 43.959999
```

4. Modify the above program as follows: The parent process should create two children.
   [User enters Value of variable ‘a’ only once]. The first child finds the area and perimeter of a circle with radius ‘a’. The Second child finds the area and perimeter of square with side ‘a’.

CODE :

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

```c
int main(){
    int a;
    printf("Enter the Value for a: ");
    scanf("%d" ,&a);

    if(!fork()){
        printf("Label -> CIRCLE PID -> %d PPID -> %d\n" , getpid() , getppid());
        printf("The area of circle is %f and perimeter is %f\n", 3.14*a*a, 2*3.14*a);
        }
    else{
        if(!fork()){
            printf("Label -> SQUARE PID -> %d PPID -> %d\n", getpid(), getppid());
            printf("Area of square is %d and perimeter is %d\n", (a*a), (4 * a));
        }
        else{wait(NULL);} // parent has to wait until both children finishes, otherwise
child will force exits with incomplete output
    }
    return 0;}
```

OUTPUT :

```
root@anuvind:~/LABSHEET4# gcc lab4q4.c -o lab4q4
root@anuvind:~/LABSHEET4# ./lab4q4
Enter the Value for a: 6
Label -> CIRCLE PID -> 636 PPID -> 635
The area of circle is 113.040000 and perimeter is 37.680000
Label -> SQUARE PID -> 637 PPID -> 635
Area of square is 36 and perimeter is 24
```

5. Modify the previous program to make the parent process wait until the completion of its children. **[Hint. Use wait() system call]**

CODE :

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

int main(){
    int a;
    printf("Enter the Value for a: ");
    scanf("%d" ,&a);

    if(!fork()){ // circle child
        printf("Label -> CIRCLE PID -> %d PPID -> %d\n" , getpid() , getppid());
        printf("The area of circle is %f and perimeter is %f\n", 3.14*a*a, 2*3.14*a);
        }
    else{
        wait(NULL);
        if(!fork()){ // square child
            printf("Label -> SQUARE PID -> %d PPID -> %d\n", getpid(), getppid());
```

```
            printf("Area of square is %d and perimeter is %d\n", (a*a), (4 * a));
        }
        else{wait(NULL);}
    } // order is always circle -> square
    return 0;}
```

OUTPUT :

```
Parent process exitingroot@anuvind:~/LABSHEET4# gcc lab4q5.c -o lab4q5
root@anuvind:~/LABSHEET4# ./lab4q5
Enter the Value for a: 5
Label -> CIRCLE PID -> 751 PPID -> 750
The area of circle is 78.500000 and perimeter is 31.400000
Label -> SQUARE PID -> 752 PPID -> 750
Area of square is 25 and perimeter is 20
```

6. Create a parent process having two children. The first child should overwrite its address space with a process that prints "Happy new year" (happynewyear.c).
   The second child should overwrite its address space with another process that prints the sum of digits of a number entered by the user(sum.c). **[Hint: use exec family of system calls]**

CODE :

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

int main(){
    if(!fork()){ // first child
        execl("./happynewyear","./happynewyear", NULL);
    }
    else{
        wait(NULL);
        if(!fork()){ // second child waits until first child finishes
            execl("./sum","./sum",NULL);
        }
        else{ // parent waits until both children finishes
            wait(NULL);
            printf("Parent process exiting ....Good bye..");

        }
    }
    return 0;
}
```

CODE (sum.c) :

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

int main(){
   printf("Please enter a number: ");
   int num;
   scanf("%d", &num);
   int sum=0;
   while(num>0){
        int k=num%10;
        sum+=k;
        num=num/10;
   }
    printf("Sum of the digits: %d\n", sum);
    return 0;
}
```

CODE (Happynewyear):

```c
#include <stdio.h>
int main(){
    printf("Happy New Year\n");
    return 0;
}
```

OUTPUT :

```
root@anuvind:~/LABSHEET4# gcc lab4q6.c -o lab4q6
root@anuvind:~/LABSHEET4# ./lab4q6
Happy New Year
Please enter a number: 4587
Sum of the digits: 24
Parent process exiting....Good bye..root@anuvind:~/LABSHEET4#
```