Amrita School of Computing, Amritapuri Campus.
19CSE213: Operating Systems

LAB SHEET 5

Process Creation using fork() System call

The following C program demonstrates the use of ***getpid () and getppid ()*** to print the PID of the process and the PID of its parent process respectively.

Header files to be included:

1. stdio.h - it is used for printf() function
2. sys/types.h - it is used for pid_t type, that is the data type of the variables which are using to store the process ids.
3. unistd.h - it is used for getpid() and getppid() functions and also for fork() system call.

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

int main(void)
{
      //variable to store calling function's process id
      pid_t process_id;
      //variable to store parent function's process id
      pid_t p_process_id;

      //getpid() - will return process id of calling
                  function
      process_id = getpid();
      //getppid() - will return process id of parent
                  function
      p_process_id = getppid();

      //printing the process ids
      printf ("The process id: %d\n",process_id);
      printf("The process id of parent function:
                %d\n", p_process_id);

      return 0;
}
```

Sample program using fork system call to create a child process

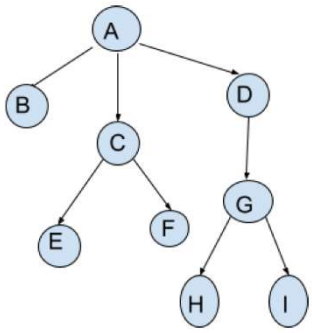
```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

int main()
{

   if (!fork())
   {
      // child process
      printf("Child process:\n");
      printf("  Process ID: %d\n", getpid());
      printf("  Parent process ID: %d\n", getppid());

   } else
   {
      // parent process
      printf("Parent process:\n");
      printf("  Process ID: %d\n", getpid());
      printf("  Parent process ID: %d\n", getppid());
      wait(NULL);
    }

   return 0;
}
```

Sample OUTPUT

```
Parent process:
  Process ID: 2664
  Parent process ID: 2600
Child process:
  Process ID: 2665
  Parent process ID: 2664
```

LAB EXERCISE

1. 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 process.



2. Execute the fork.c program given to you more than once. What is the order in which the processes are being executed? Is it the same in every execution?
3. Modify the fork.c program using **wait () or sleep ()** system call, so that parent will wait until child completes its execution.
4. Write a program to find area and perimeter of circle and square. Create separate processes for circle and square.
5. Modify the above program as follows: Parent process should create two children.
   [**User enters Value of variable 'a' only once**]
   The first child finds area and perimeter of a circle with radius 'a'
   The Second child finds area and perimeter of square with side 'a'.
6. Modify the previous program to make the parent process wait until the completion of its children.
7. Modify question 5 to make the parent process wait until the completion of its child process that finds area of the Square.
8. Create a parent process having two children. The first child should overwrite its address space with a process that prints "Happy new year". The second child should overwrite its address space with another process that prints the sum of digits of a number entered by the user. **[Hint: use exec family of system calls]**
   Sample output: The output should come in the following order

   Happy new year

   Enter the number: 123

Sum of Digits: 6

Parent exiting …good bye.