CSE 4049: Design of Operating Systems

ASSIGNMENT 1:

This assignment is designed to give you practice with the concepts of

- Process management
  - Trace different states of the process during execution
  - To learn the use of different system calls such as (fork(),vfork(),wait(), exit()) for process handling in Linux environment.

1. Write a C program to create a child process using fork() system call. The child process will print the message “Child” with its process identifier and then continue in an indefinite loop. The parent process will print the message “Parent” with its process identifier and then continue in an indefinite loop.
   a) Run the program and trace the state of both processes.
   b) Terminate the child process. Then trace the state of processes.
   c) Run the program and trace the state of both processes. Terminate the parent process. Then trace the state of processes.
   d) Modify the program so that the parent process after displaying the message will wait for child process to complete its task. Again run the program and trace the state of both processes.
   e) Terminate the child process. Then trace the state of processes.

2. Trace the output of the following program segment :

a) int main( )
{
if(fork()==0)
printf("1");
else
printf("2");
printf("3");
return 0;
 	
}
	b) int main( )
	{
	if(vfork()==0)
	printf("1");
	else
	printf("2");
	printf("3");
	}

c) int main( )
{
pid_t pid;
int i=5;
pid=fork();
i=i+1;
if(pid= =0)
{
printf("Child: %d",i);
}
else
	d) int main( )
	{
	pid_t pid;
	int i=5;
	pid=vfork();
	i=i+1;
	if(pid==0)
	{
	printf("Child: %d",i);
	_exit(0);
	}

```
      {
      wait(NULL);
      printf("Parent: %d",i);
      }
     return 0;
      }
```

```
      else
      {
      printf("Parent: %d",i);
      }
    return 0;
      }
```

```
e)  int main( )
      {
      pid_t pid;
      int i=5;
      pid=fork();
      if(pid= =0)
      {
      i=i+1;
      printf("Child: %d",i);
      }
      else
      {
      wait(NULL);
      printf("Parent: %d",i);
      }
      return 0;
      }
```

```
f)  int main( )
      {
      pid_t pid;
      int i=5;
      pid=vfork();
      if(pid==0)
      {
      i=i+1;
      printf("Child: %d",i);
      _exit(0);
      }
      else
      {
      printf("Parent: %d",i);
      }
    return 0;
      }
```

```
g)  int main( )
      {
      int i=5;
      if(fork( )==0)
      {
      printf("Child: %d",i);
      }
      else
      {
      printf("Parent: %d",i);
      }
      return 0;
      }
```

```
h)  int main( )
      {
      int i=5;
      if(vfork( )==0)
      {
      printf("Child: %d",i);
      _exit(0);
      }
      else
      {
      printf("Parent: %d",i);
      }
      return 0;
      }
```

```
i)  int main( )
      {
      if(fork( )==0)
      {
      printf("1");
      }
      else
      {
      wait(NULL);
      printf("2");
      printf("3");
      }
      return 0;
      }
```

```
j)  int main( )
      {
      if(vfork( )==0)
      {
      printf("1");
      _exit(0);
      }
      else
      {
      printf("2");
      printf("3");
      }
     return 0;
      }
```

```
k)  int main( )
        {
        pid_t c1;
        int n=10;
        c1=fork( );
        if(c1==0)
        {
        printf(" Child\n");
        n=20;
        printf("n=%d \n",n);
        }
        else
        {
        wait(NULL);
        printf("Parent\n");
        printf("n=%d \n",n);
        }
        return 0;
        }
```

```
l)  int main( )
        {
        pid_t c1;
        int n=10;
        c1=vfork( );
        if(c1==0)
        {
        printf(" Child\n");
        n=20;
        printf("n=%d \n",n);
        _exit(0);
        }
        else
        {
        printf("Parent\n");
        printf("n=%d \n",n);
        }
       return 0;
       }
```

```
m)  int main( )
        {
        int i=5;
        fork();
        i=i+1;
        fork();
        fprintf ( stderr,"% d",i);
        return 0;
        }
```

```
n)  int main( )
        {
        pid_t pid;
        int i=5;
        pid=vfork();
        if(pid==0)
        {
        printf("Child: %d",i);
        _exit(0);
        }
        else
        {
        i=i+1;
        printf("Parent: %d",i);
        }
        return 0;
        }
```

```
o)  int main( )
        {
        int i=5;
        if(fork()==0)
        i=i+1;
        else
        i=i-1;
        fprintf(stderr,"%d",i);
        return 0;
        }
```

```
p)  int main( )
        {
        int i=5;
        if(vfork()==0)
        {
        i=i+1;
        _exit(0);
        }
        else
        i=i-1;
        fprintf(stderr,"%d",i);
        return 0;
        }
```

```
q)  int main( )
        {
        int j,i=5;
        for(j=1;j<3;j++)
        {
        if(fork()==0)
        {
        i=i+1;
        break;
        }
        else
        wait(NULL);
        }
        fprintf(stderr,"%d",i);
        return 0;
        }
```

```
r)  int main( )
        {
        int j,i=5;
        for(j=1;j<3;j++)
        {
        if(fork()!=0)
        {
        i=i-1;
        break;
        }
        }
        fprintf(stderr,"%d",i);
        return 0;
        }
```

```
s)  int main( )
        {
         if(fork() == 0)

           if(fork())

             printf("1\n");

         return 0;

        }
```

```
t)  void fun1(){

            fork();

            fork();

            printf("1\n");

    }

    int main() {

            fun1();

            printf("1\n");

            return 0;

    }
```

3. Trace the following program segment and determine how many processes are created. Draw a graph that shows how the processes are related.

```
a.  int main( )
       {
              if(fork( )&&fork( ));
              printf(" 1”);
              return 0;
       }
```

```
b.  int main( )
       {
              if(fork( ) || fork( ));
              printf(" 1”);
              return 0;
       }
```

```
c.  int main( )
       {
              pid_t c1,c2;
               c2=0;
```

```
d.  int main( )
       {
              pid_t c1=1,c2=1;
                c1 = fork();
```

```
            c1 = fork();
            if (c1 == 0)
               c2 = fork();
            if (c2 > 0)
               fork();
         printf(" 1”);
         return 0;
}
```

```
            if (c1 != 0)
               c2 = fork();
            if (c2== 0)
                 fork();
         printf(" 1”);
         return 0;
}
```

```
e.  int main( )
        {
           if (fork() || fork())
               fork();
           printf(" 1 ");
           return 0;
        }
```

```
f.  int main( )
        {
           if (fork() && (!fork()))
           {
              if (fork() || fork())
              {
                 fork();
              }
           }
           printf("2 ");
           return 0;
        }
```