PROCESS ASSIGNMENT

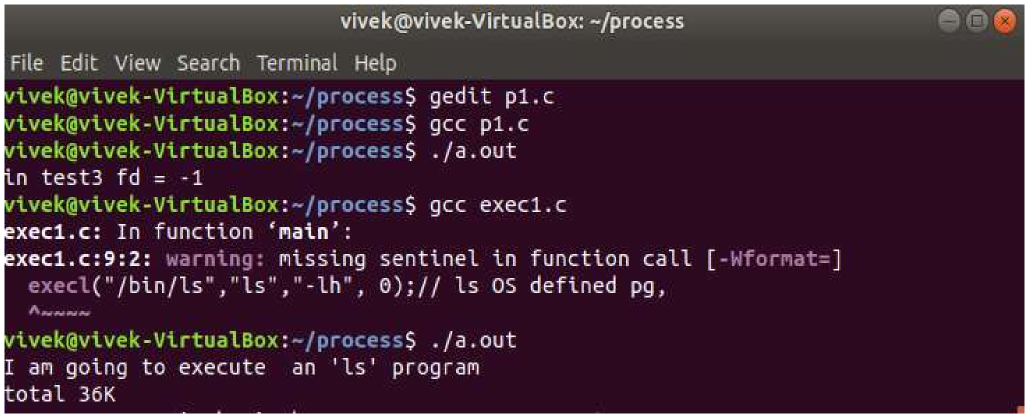
1. **Test whether the process(exec() system call) that replaces old program data , will inherit the fds or not.**

Code :

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

int main()
{
        int fd = open("./exec1",O_RDONLY);
        printf("in test3 fd = %d\n",fd);
}
```

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

int main()
{
            printf("I am going to execute  an 'ls' program\n");

            execl("/bin/ls","ls","-lh", 0);

            printf("i executed ls program ");
            printf("i executed ls program ");
            printf("i executed ls program ");
}
```

Output:


vivek@vivek-VirtualBox: ~/process
File Edit View Search Terminal Help

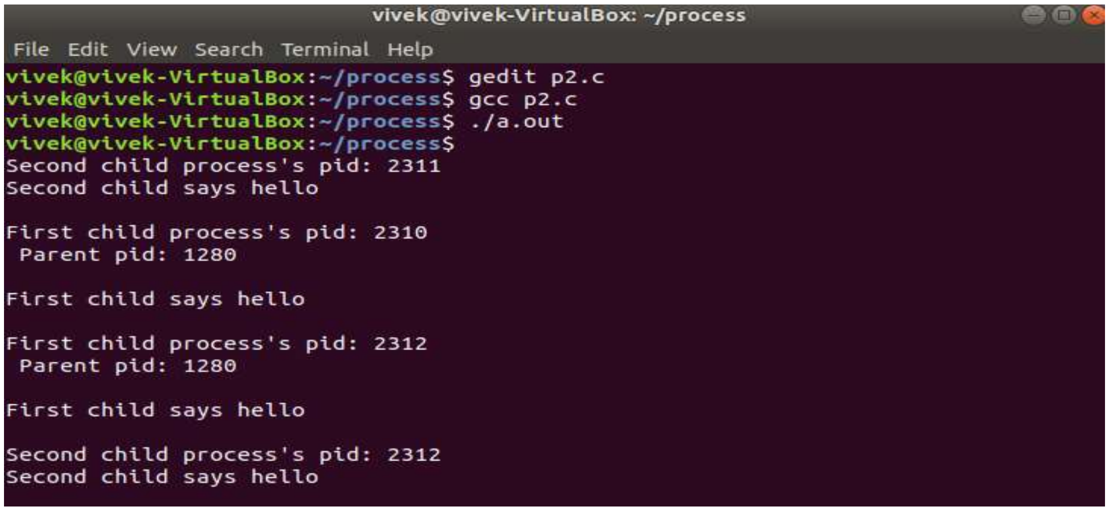
```
vivek@vivek-VirtualBox:~/process$ gedit p1.c
vivek@vivek-VirtualBox:~/process$ gcc p1.c
vivek@vivek-VirtualBox:~/process$ ./a.out
in test3 fd = -1
vivek@vivek-VirtualBox:~/process$ gcc exec1.c
exec1.c: In function ‘main’:
exec1.c:9:2: warning: missing sentinel in function call [-Wformat=]
  execl("/bin/ls","ls","-lh", 0);// ls OS defined pg,
  ^~~~~
vivek@vivek-VirtualBox:~/process$ ./a.out
I am going to execute  an 'ls' program
total 36K
```

2. **Write a program such that parent process create two child processes,such that each child executes a separate task.**

Code:

```
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>
int main()
{
int p1,p2;
p1 = fork();
p2 = fork();
if(p1 == 0)
        {
                printf("\nFirst child process's pid: %d\n Parent pid: %d\n",getpid(),getppid());
                printf("\nFirst child says hello\n");
        }
if(p2 == 0)
        {
                printf("\nSecond child process's pid: %d",getpid());
                printf("\nSecond child says hello\n");
        }
return 0;
}
```

Output:


vivek@vivek-VirtualBox: ~/process
File Edit View Search Terminal Help

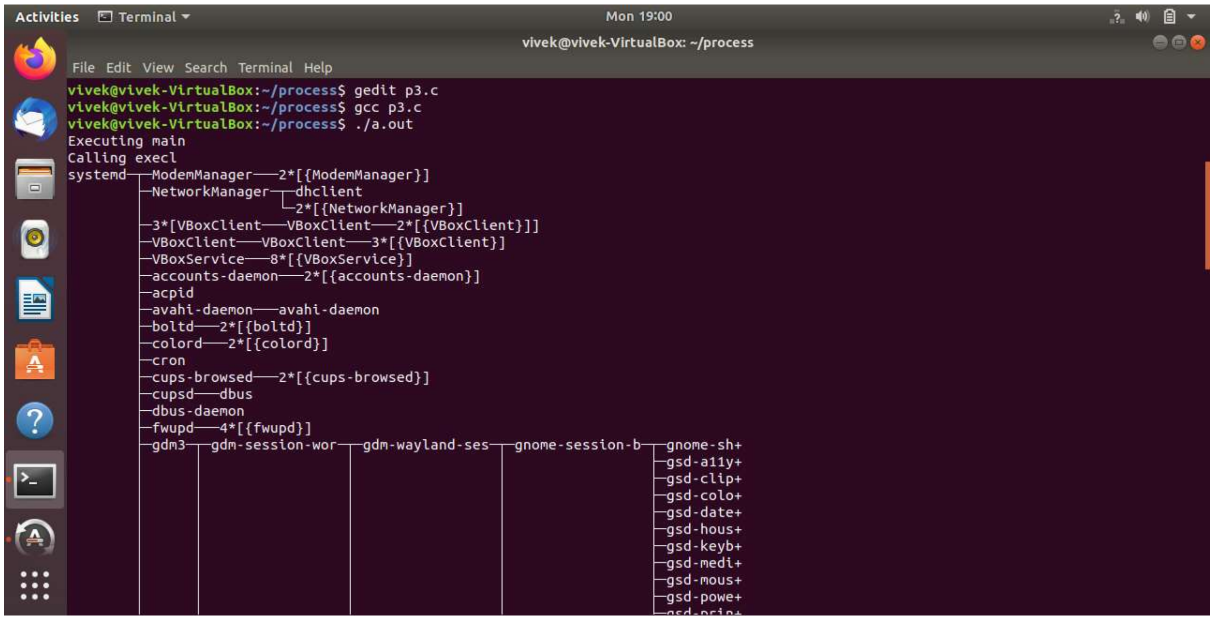
```
vivek@vivek-VirtualBox:~/process$ gedit p2.c
vivek@vivek-VirtualBox:~/process$ gcc p2.c
vivek@vivek-VirtualBox:~/process$ ./a.out
vivek@vivek-VirtualBox:~/process$
Second child process's pid: 2311
Second child says hello

First child process's pid: 2310
 Parent pid: 1280

First child says hello

First child process's pid: 2312
 Parent pid: 1280

First child says hello

Second child process's pid: 2312
Second child says hello
```

3. **A program that replaces old program with new program data and is expected to display the currently running processes in a hierarchical tree format.**

Code:

```
#include <stdio.h>
#include <unistd.h>
int main()
{
printf("Executing main\nCalling execl\n");
execl("/usr/bin/pstree", "pstree", NULL);
return 0;
}
```

Output:



Activities
Terminal
Mon 19:00
vivek@vivek-VirtualBox: ~/process
File Edit View Search Terminal Help

```
vivek@vivek-VirtualBox:~/process$ gedit p3.c
vivek@vivek-VirtualBox:~/process$ gcc p3.c
vivek@vivek-VirtualBox:~/process$ ./a.out
Executing main
Calling execl
systemd─┬─ModemManager───2*[{ModemManager}]
        ├─NetworkManager─┬─dhclient
        │                └─2*[{NetworkManager}]
        ├─3*[VBoxClient───VBoxClient───2*[{VBoxClient}]]
        ├─VBoxClient───VBoxClient───3*[{VBoxClient}]
        ├─VBoxService───8*[{VBoxService}]
        ├─accounts-daemon───2*[{accounts-daemon}]
        ├─acpid
        ├─avahi-daemon───avahi-daemon
        ├─boltd───2*[{boltd}]
        ├─colord───2*[{colord}]
        ├─cron
        ├─cups-browsed───2*[{cups-browsed}]
        ├─cupsd───dbus
        ├─dbus-daemon
        ├─fwupd───4*[{fwupd}]
        ├─gdm3─┬─gdm-session-wor─┬─gdm-wayland-ses─┬─gnome-session-b─┬─gnome-sh+
        │      │                 │                 │                 ├─gsd-a11y+
        │      │                 │                 │                 ├─gsd-clip+
        │      │                 │                 │                 ├─gsd-colo+
        │      │                 │                 │                 ├─gsd-date+
        │      │                 │                 │                 ├─gsd-hous+
        │      │                 │                 │                 ├─gsd-keyb+
        │      │                 │                 │                 ├─gsd-medi+
        │      │                 │                 │                 ├─gsd-mous+
        │      │                 │                 │                 ├─gsd-powe+
```

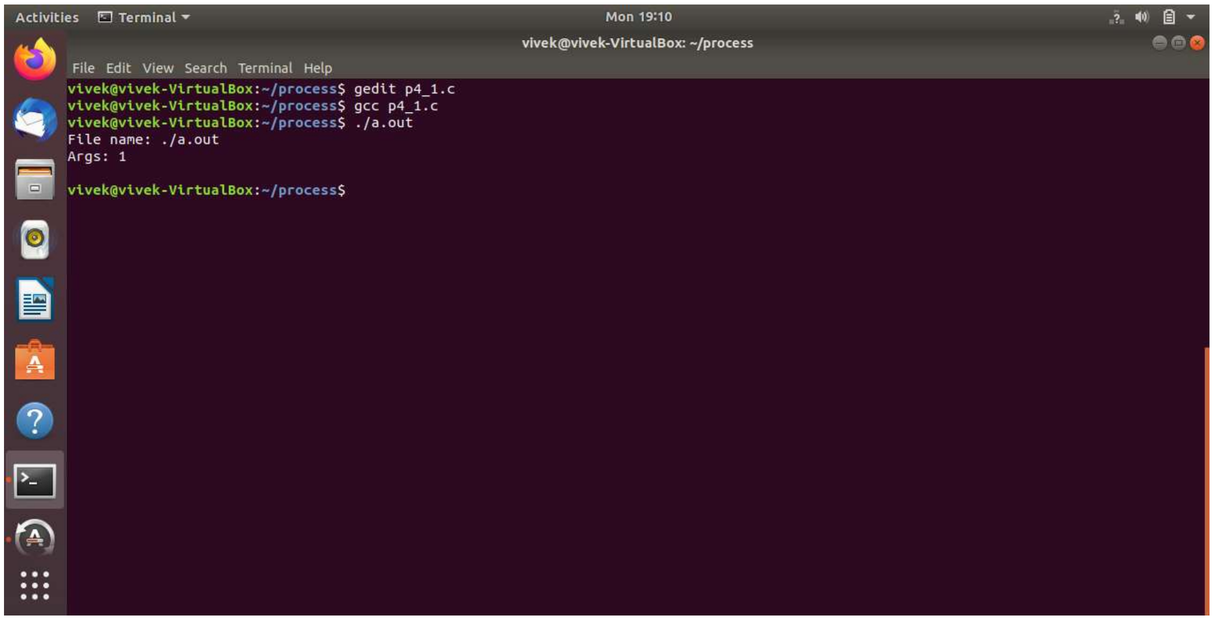
4. **A process using execl should replace a new command line program.**

Code:

```
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>
int main()
{
int a,b;
a = fork();
if(a == 0)
{
printf("Child pid: %d\n",getpid());
execl("k", "./k", "a b c d", "e f g h" ,NULL);
}
else
{
printf("Parent waiting for pid: %d\n",waitpid(a,&b,0));
}
return 0;
}
```

```
#include <stdio.h>
#include <sys/wait.h>
#include <stdlib.h>
int main(int argc, char *argv[])
{
        int i;
        printf("File name: %s\n", argv[0]);
        printf("Args: %d\n",argc);
        for(i=1;i<argc;i++)
        {
                printf("%s",argv[i]);
        }
        printf("\n");
        return 0;
}
```

Output:


Activities
Terminal
Mon 19:10
vivek@vivek-VirtualBox: ~/process
File Edit View Search Terminal Help

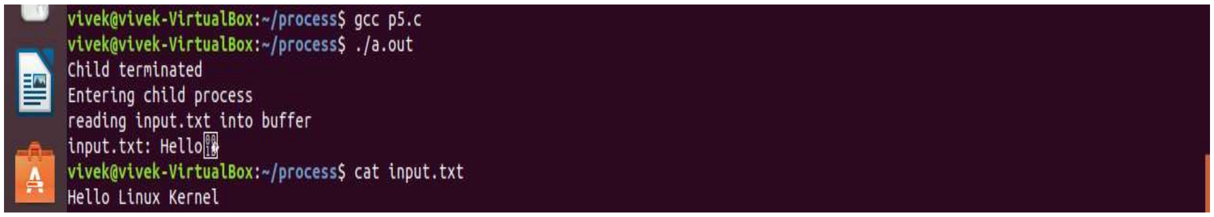
```
vivek@vivek-VirtualBox:~/process$ gedit p4_1.c
vivek@vivek-VirtualBox:~/process$ gcc p4_1.c
vivek@vivek-VirtualBox:~/process$ ./a.out
File name: ./a.out
Args: 1

vivek@vivek-VirtualBox:~/process$
```

5. Write a program where parent process waits until child process opens and reads a file into an empty buffer.

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/wait.h>
int main()
{
int f1, c1;
c1 = fork();
char c[10];
if(c1 == 0)
{
	printf("Entering child process\nreading input.txt into buffer\n");
	f1 = open("input.txt",O_RDONLY,NULL);
	read(f1,c,5);
	printf("input.txt: %s\n",c);
	close(f1);
}
else
{
	printf("Child terminated\n");
	wait(NULL);
}
return 0;
}
```

Output:

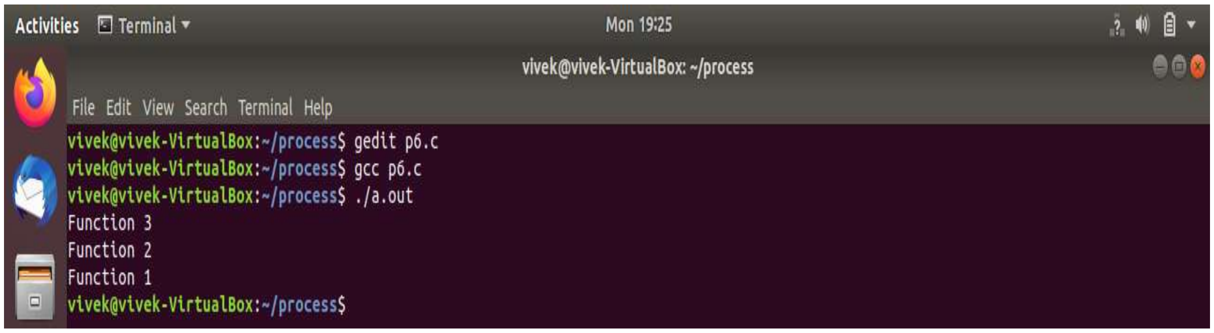


```
vivek@vivek-VirtualBox:~/process$ gcc p5.c
vivek@vivek-VirtualBox:~/process$ ./a.out
Child terminated
Entering child process
reading input.txt into buffer
input.txt: Hello
vivek@vivek-VirtualBox:~/process$ cat input.txt
Hello Linux Kernel
```

6. **Write a program where functions of the program are called in reverse order of their function calls from main.**
   Code:

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

inline void print1();
inline void print2();
inline void print3();
int main()
{
	atexit(print1);
	atexit(print2);
	atexit(print3);
	return 0;
}

void print1()
{
	printf("Function 1\n");
}
void print2()
{
	printf("Function 2\n");
}
void print3()
{
	printf("Function 3\n");
}
```

OUTPUT:


Activities
Terminal
Mon 19:25
vivek@vivek-VirtualBox: ~/process
File Edit View Search Terminal Help

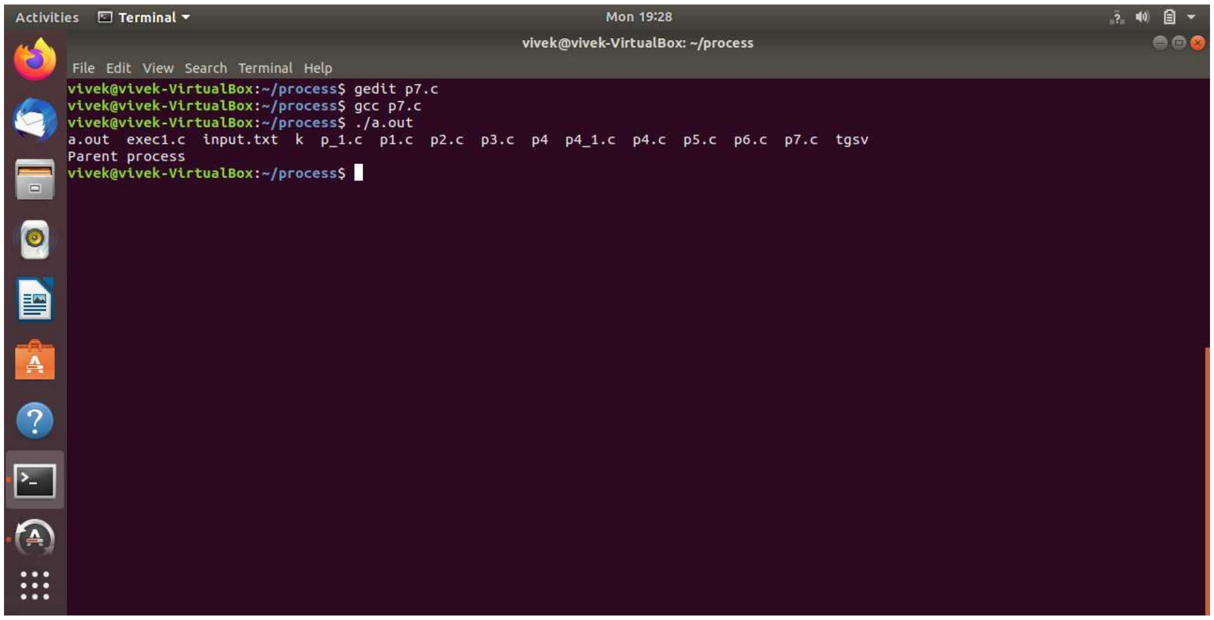
```
vivek@vivek-VirtualBox:~/process$ gedit p6.c
vivek@vivek-VirtualBox:~/process$ gcc p6.c
vivek@vivek-VirtualBox:~/process$ ./a.out
Function 3
Function 2
Function 1
vivek@vivek-VirtualBox:~/process$
```

7. **Write a program where child executes new execl program while parent waits for child task to complete.**

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>
int main() {
int c,status;
c = fork();
if(c == 0) {
execl("/bin/ls","ls",NULL);
}
waitpid(c,&status,0);
printf("Parent process\n");
return 0;
}
```

OUTPUT:


Activities
Terminal
Mon 19:28
vivek@vivek-VirtualBox: ~/process
File Edit View Search Terminal Help

```
vivek@vivek-VirtualBox:~/process$ gedit p7.c
vivek@vivek-VirtualBox:~/process$ gcc p7.c
vivek@vivek-VirtualBox:~/process$ ./a.out
a.out  exec1.c  input.txt  k  p_1.c  p1.c  p2.c  p3.c  p4  p4_1.c  p4.c  p5.c  p6.c  p7.c  tgsv
Parent process
vivek@vivek-VirtualBox:~/process$
```

GITHUB LINK :

https://github.com/VIVEK0014/Linux_internals/tree/main/Process_Assignment