Linux Internals
Assignment 2
By: Pavan Hegde

program 1:

Test whether the process (exec() system call)that replaces old program, will inherit the fd's or not.

code:

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

int main()
{
        int fd;
        fd=open("execl.c",O_RDONLY,777);
        printf("fd for execl is = %d\n",fd);
        close(fd);
        return 0;
}
```

output:

```
pavan@pavan-VirtualBox:~/Training/Linux_internals_tools/Day4/Assig$ gedit prog2.c
pavan@pavan-VirtualBox:~/Training/Linux_internals_tools/Day4/Assig$ gcc -o p2 prog2.c
pavan@pavan-VirtualBox:~/Training/Linux_internals_tools/Day4/Assig$ ./p2
fd for execl is = 3
```

Program 2:

write a program such that parent process create two child processes.

code:

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

int main()
{
        int pid,pid1;
        printf("current process id = %d\n",getpid());
        pid=fork();
        while(1){
        if(pid==0)
        {
                printf("child process id 1= %d\n",getpid());
                printf("%d\n",getppid());
        }
        else
        {
```

```
                pid1=fork();
                if(pid1==0)
                {
                        printf("\nchild process id 2= %d\n",getpid());
                        printf("%d\n",getppid());
                }
                else
                {
                        printf("parent process id = %d\n",getpid());
                }
        }
        exit(0);
        }
        return 0;
}
```

Output:

```
pavan@pavan-VirtualBox:~/Training/Linux_internals_tools/Day4/Assig$ gedit prog1.c
pavan@pavan-VirtualBox:~/Training/Linux_internals_tools/Day4/Assig$ gcc prog1.c
pavan@pavan-VirtualBox:~/Training/Linux_internals_tools/Day4/Assig$ ./a.out
current process id = 2077
parent process id = 2077
pavan@pavan-VirtualBox:~/Training/Linux_internals_tools/Day4/Assig$
child process id 2= 2079
1290
child process id 1= 2078
1290
```

program 3:

A program that replace old program with vim editor program and open a new text file.

code:

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

int main()
{
        int x;
        x=execl("/usr/bin/vi","vim","info.txt",0);
        if(x==-1)
                printf("error creating file %d\n",x);
        return 0;
}
```

output:

it will open the info.txt file

program 4:

a process using execl() system call should replace a new command line program.

code:

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

int main(int argc,char *argv[])
{
        int i;
        printf("\n file name : %s\n",argv[0]);
        printf("\n total number of arguments : %d\n",argc);
        execl("prog1","./prog1","linux","kernel","programming","device","drivers",0);
        printf("\n Arguments passed: ");
        for(i=1;i<argc;i++)
        {
                printf("%s",argv[i]);
        }
        printf("\n");

        return 0;
}
```

output:

```
pavan@pavan-VirtualBox:~/Training/Linux_internals_tools/Day4/Assig$ gedit prog4.c
pavan@pavan-VirtualBox:~/Training/Linux_internals_tools/Day4/Assig$ gcc -o prog4 prog4.c
pavan@pavan-VirtualBox:~/Training/Linux_internals_tools/Day4/Assig$ ./prog4

 file name : ./prog4

 total number of arguments : 1
current process id = 10416
parent process id = 10416
pavan@pavan-VirtualBox:~/Training/Linux_internals_tools/Day4/Assig$
child process id 2= 10418
1290
child process id 1= 10417
1290
```

program 5:

write a program parent process wait untill,while child process open file and read file data into empty buffer.

code:

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

int main()
{
	int fd;
	char buff1[100]="My name is pavan,byee";
	char buff2[100];
	pid_t pid;
	pid=fork();
	if(pid==0)
	{
		sleep(5);
		printf("I am child with delay of 5 sec & my child pid = %d\n",getpid());
		fd=open("seek_set.txt",O_CREAT | O_RDWR,777);
		printf("fd = %d\n",fd);
		if(fd>0)
		{
			write(fd,buff1,100);
		}
		else
		{
			printf("error not created seek_set.txt\n");
		}
		lseek(fd,0,SEEK_SET);
		read(fd,buff2,100);
		printf("Data is written in seek_set.txt is ---  %s\n",buff2);
	}
	else
	{
		wait(0);
		printf("I am parent process pid = %d\n",getpid());
	}
	return 0;
}
```

output:

```
pavan@pavan-VirtualBox:~/Training/Linux_internals_tools/Day4/Assig$ gedit prog5.c
pavan@pavan-VirtualBox:~/Training/Linux_internals_tools/Day4/Assig$ gcc prog5.c
pavan@pavan-VirtualBox:~/Training/Linux_internals_tools/Day4/Assig$ ./a.out
I am child with delay of 5 sec & my child pid = 5433
fd = 3
Data is written in seek_set.txt is ---  My name is pavan,byee
I am parent process pid = 5432
```

program 6:

write a program,where functions of the programare called in the reverse order of theirfunction calls from main().

code:

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


void callback1(void)
{
        printf("call back 1 func called\n");
}

void callback2(void)
{
        printf("call back 2 func called\n");
}

void callback3(void)
{
        printf("call back 3 func called\n");
}


int main()
{
        printf("registering callback1\n");
        atexit(callback1);
        printf("registering callback2\n");
        atexit(callback2);
        printf("registering callback3\n");
        atexit(callback3);
        printf("main exiting now....\n");
        exit(0);
}
```

output:

```
pavan@pavan-VirtualBox:~/Training/Linux_internals_tools/Day4/Assig$ gedit prog6.c
pavan@pavan-VirtualBox:~/Training/Linux_internals_tools/Day4/Assig$ gcc -o prog6 prog6.c
pavan@pavan-VirtualBox:~/Training/Linux_internals_tools/Day4/Assig$ ./prog6
registering callback1
registering callback2
registering callback3
main exiting now....
call back 3 func called
call back 2 func called
call back 1 func called
```

program 7:

write a program child executes(exec())a new program,while parent waitsfor child task to get complete.

code:

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

int main()
{
        pid_t pid;
        pid=fork();
        if(pid==0)
        {
                printf("my child pid = %d\n",getpid());
        }
        else
        {
                int pid1;
                pid1=wait(0);
                printf("I am parent process pid = %d\n",getpid());
        }
        return 0;
}
```

output:

```
pavan@pavan-VirtualBox:~/Training/Linux_internals_tools/Day4/Assig$ gedit prog7.c
pavan@pavan-VirtualBox:~/Training/Linux_internals_tools/Day4/Assig$ gcc -o prog7 prog7.c
pavan@pavan-VirtualBox:~/Training/Linux_internals_tools/Day4/Assig$ ./prog7
my child pid = 10519
I am parent process pid = 10518
```