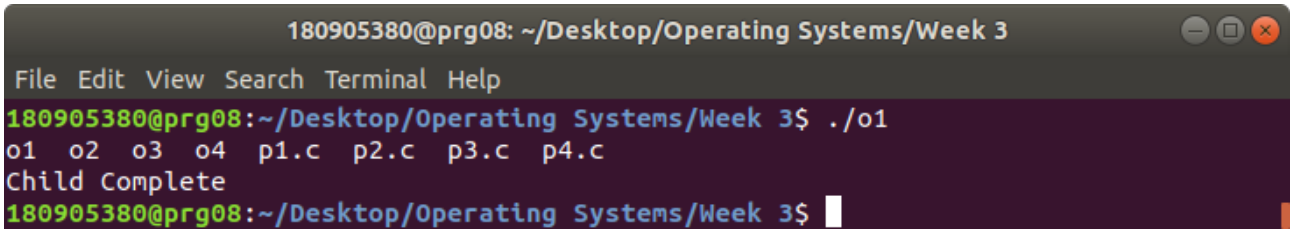
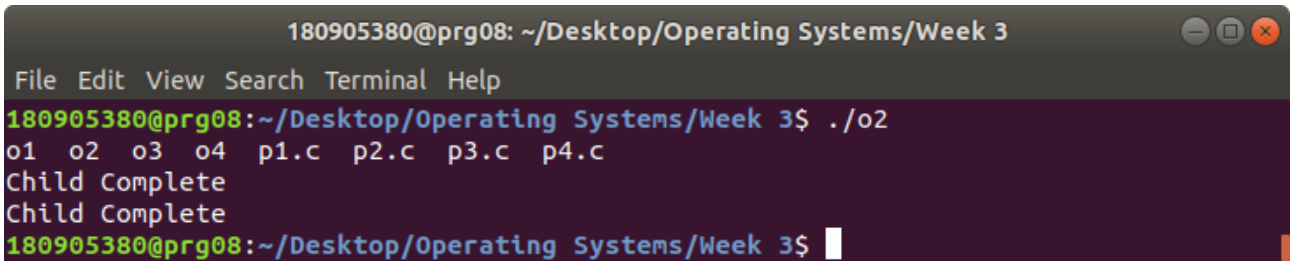
Q1. Write a C program to block a parent process until the child completes using a wait system call.

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

int main()
{
	pid_t pid;
	pid = fork();
	if (pid < 0)
	{
		fprintf(stderr,"Fork Failed");
		exit(-1);
	}
	else if (pid == 0)
	{
		execlp("/bin/ls","ls",NULL);
	}
	else
	{
		wait(NULL);
		printf("Child Complete\n");
		exit(0);
	}
}
```



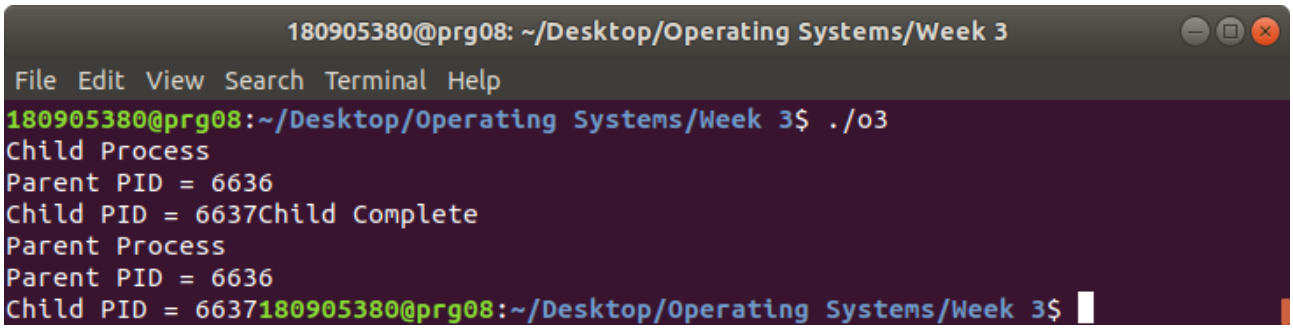
Q2. Write a C program to load the binary executable of the previous program in a child process using the exec system call.

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

int main()
{
	pid_t pid;
	pid = fork();
	if (pid < 0)
	{
		fprintf(stderr,"Fork Failed");
		exit(-1);
	}
	else if (pid == 0)
	{
		execlp("./o1","o1",NULL);
	}
	else
	{
		wait(NULL);
		printf("Child Complete\n");
		exit(0);
	}
}
```



Q3. Write a program to create a chiled process. Display the process IDs of the process, parent and child (if any) in both parent and child processes.

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

int main()
{
	pid_t pid;
	pid = fork();
	if (pid < 0)
	{
		fprintf(stderr,"Fork Failed");
		exit(-1);
	}
	else if (pid == 0)
	{
		printf("Child Process\n");
		pid_t cpid,ppid;
		cpid = getpid();
		ppid = getppid();
		printf("Parent PID = %d\nChild PID = %d",ppid,cpid);
	}
	else
	{
		wait(NULL);
		printf("Child Complete\n");
		printf("Parent Process\n");
		pid_t ppid;
		ppid = getpid();
		printf("Parent PID = %d\nChild PID = %d",ppid,pid);
		exit(0);
	}
}
```



Q4. Create a zombie(defunct) child process (a chiled with exit() call, but no corresponding wait() in the sleeping parent) and allow init process to adopt it (after parent terminates). Run the process as a background process and run the “ps” command.

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

int main()
{
	pid_t pid;
	pid = fork();
	if (pid < 0)
	{
		fprintf(stderr,"Fork Failed");
		exit(-1);
	}
	else if (pid == 0)
	{
		pid_t cpid;
		cpid = fork();
		if (cpid < 0)
		{
			fprintf(stderr,"Fork Failed");
			exit(-1);
		}
		else if (cpid == 0)
		{
			sleep(20);
			printf("Grandchild Process\n");
			exit(0);
		}
		else
		{
			printf("Child Process\n");
			exit(0);
		}
	}
	else
	{
		sleep(30);
		printf("Parent Process\n");
		exit(0);
	}
}
```

```
180905380@prg08: ~/Desktop/Operating Systems/Week 3
File  Edit  View  Search  Terminal  Help
180905380@prg08:~/Desktop/Operating Systems/Week 3$ cc -o o4 p4.c
180905380@prg08:~/Desktop/Operating Systems/Week 3$ ./o4 &
[1] 7592
180905380@prg08:~/Desktop/Operating Systems/Week 3$ Child Process
ps
  PID TTY          TIME CMD
 4958 pts/0    00:00:00 sh
 4963 pts/0    00:00:00 bash
 7592 pts/0    00:00:00 o4
 7593 pts/0    00:00:00 o4 <defunct>
 7594 pts/0    00:00:00 o4
 7595 pts/0    00:00:00 ps
180905380@prg08:~/Desktop/Operating Systems/Week 3$ Grandchild Process
ps
  PID TTY          TIME CMD
 4958 pts/0    00:00:00 sh
 4963 pts/0    00:00:00 bash
 7592 pts/0    00:00:00 o4
 7593 pts/0    00:00:00 o4 <defunct>
 7599 pts/0    00:00:00 ps
180905380@prg08:~/Desktop/Operating Systems/Week 3$ Parent Process

[1]+  Done                    ./o4
180905380@prg08:~/Desktop/Operating Systems/Week 3$
```