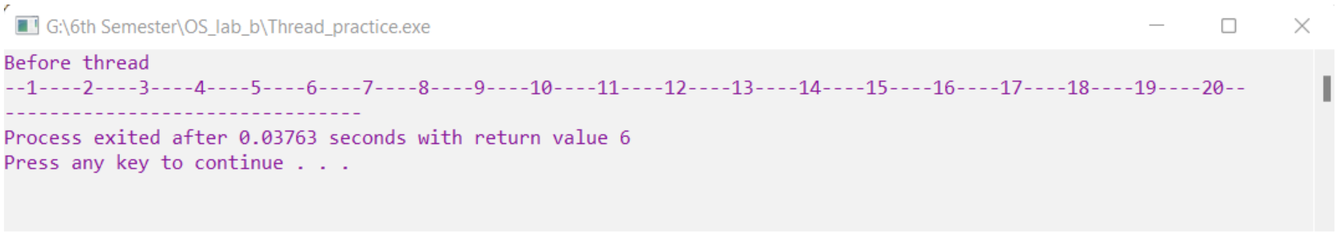
Problem 1: Thread Creation.

Code:

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

void *thread_create(void *args){
	for(int i=1;i<=20;i++){
		printf("--%d--",i);
	}
}

int main(){
	pthread_t t1;
	printf("Before thread\n");
	pthread_create(&t1,NULL,thread_create,NULL);
	//pthread_join(t1,NULL);
	pthread_exit(0);
	return 0;

}
```

Output:

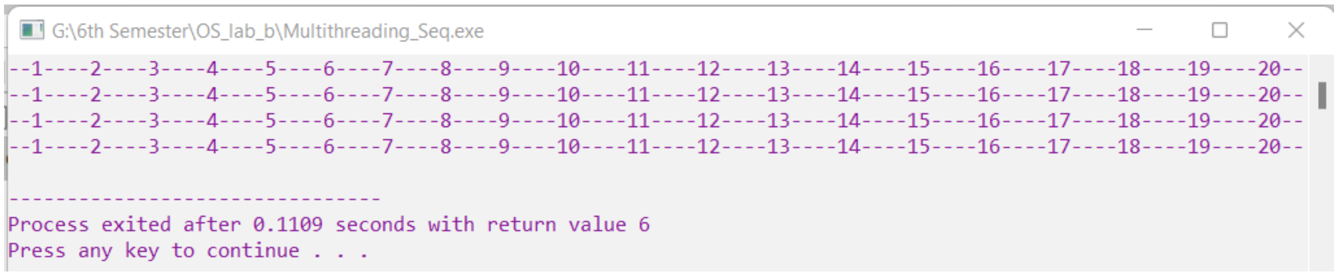


```
G:\6th Semester\OS_lab_b\Thread_practice.exe
Before thread
--1----2----3----4----5----6----7----8----9----10----11----12----13----14----15----16----17----18----19----20--
--------------------------------
Process exited after 0.03763 seconds with return value 6
Press any key to continue . . .
```

Problem 2: Multithreading sequential.

Code:

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

void threadFunc(void *args){
        for(int i=1;i<=20;i++){
                printf("--%d--",i);
        }
        return 0;
}

void main(){
        pthread_t t[5];
        for(int i=0;i<5;i++){
                int rc=pthread_create(t,NULL,threadFunc,(void *)&i);
                pthread_join(t[i],NULL);
        }
        pthread_exit(NULL);
        return 0;
}
```



G:\6th Semester\OS_lab_b\Multithreading_Seq.exe

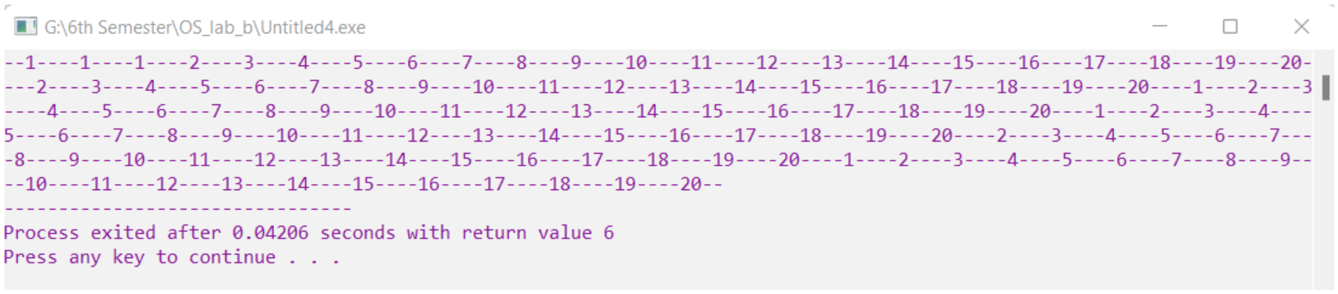
```
--1----2----3----4----5----6----7----8----9----10----11----12----13----14----15----16----17----18----19----20--
--1----2----3----4----5----6----7----8----9----10----11----12----13----14----15----16----17----18----19----20--
--1----2----3----4----5----6----7----8----9----10----11----12----13----14----15----16----17----18----19----20--
--1----2----3----4----5----6----7----8----9----10----11----12----13----14----15----16----17----18----19----20--

--------------------------------
Process exited after 0.1109 seconds with return value 6
Press any key to continue . . .
```

I used pthread_join(). So other threads wait for current thread.

Problem 3: Multithreading parallel.

Code:

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

void ThreadFunc(void *arg){
	for(int i=1;i<=20;i++){
		printf("--%d--",i);
	}
	return 0;
}

void main(){
	pthread_t t[5];
	for(int i=0;i<=5;i++){
		int rc=pthread_create(t,NULL,ThreadFunc,(void *)&i);
	}
	pthread_exit(NULL);
	return 0;
}
```



G:\6th Semester\OS_lab_b\Untitled4.exe

```
--1----1----1----2----3----4----5----6----7----8----9----10----11----12----13----14----15----16----17----18----19----20-
---2----3----4----5----6----7----8----9----10----11----12----13----14----15----16----17----18----19----20----1----2----3
----4----5----6----7----8----9----10----11----12----13----14----15----16----17----18----19----20----1----2----3----4----
5----6----7----8----9----10----11----12----13----14----15----16----17----18----19----20----2----3----4----5----6----7---
-8----9----10----11----12----13----14----15----16----17----18----19----20----1----2----3----4----5----6----7----8----9--
--10----11----12----13----14----15----16----17----18----19----20--
--------------------------------
Process exited after 0.04206 seconds with return value 6
Press any key to continue . . .
```

Problem 4: Producer Consumer Problem.

Theory:

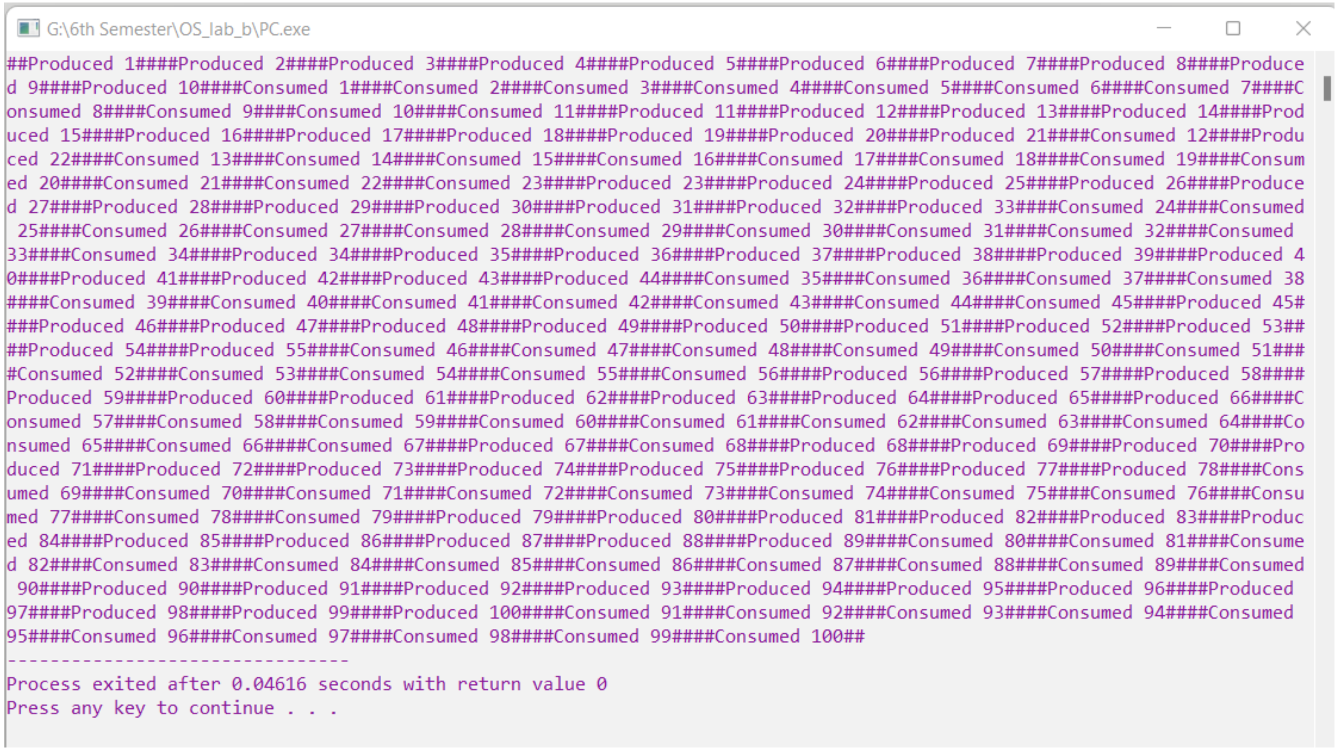
The producer-consumer problem is an example of a multi-process synchronization problem. The problem describes two processes, the producer and the consumer that shares a common fixed-size buffer use it as a queue.

- •The producer’s job is to generate data, put it into the buffer, and start again.
- •At the same time, the consumer is consuming the data (i.e., removing it from the buffer), one piece at a time.

Code:

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

int used=0;
int buffer[10];
int size=10;

void *Producer(void *args){
        int i;
        while(i<100){
                if(used<size){
                        buffer[i%size]=i+1;
                        used=used+1;
                        i=i+1;
                        printf("##Produced %d##",i);
                }
        }
}

void *Consumer(void *args){
        int i;
        while(i<100){
                if(used>0){
                        printf("##Consumed %d##",buffer[i%size]);
                        i=i+1;
                        used=used-1;
                }
        }
}

void main(){
        pthread_t t1,t2;
        pthread_create(&t1,NULL,Producer,NULL);
        pthread_create(&t2,NULL,Consumer,NULL);
        pthread_join(t1,NULL);
        pthread_join(t2,NULL);
        pthread_exit(0);

}
```

Output:



G:\6th Semester\OS_lab_b\PC.exe

```
##Produced 1####Produced 2####Produced 3####Produced 4####Produced 5####Produced 6####Produced 7####Produced 8####Produce
d 9####Produced 10####Consumed 1####Consumed 2####Consumed 3####Consumed 4####Consumed 5####Consumed 6####Consumed 7####C
onsumed 8####Consumed 9####Consumed 10####Consumed 11####Produced 11####Produced 12####Produced 13####Produced 14####Prod
uced 15####Produced 16####Produced 17####Produced 18####Produced 19####Produced 20####Produced 21####Consumed 12####Produ
ced 22####Consumed 13####Consumed 14####Consumed 15####Consumed 16####Consumed 17####Consumed 18####Consumed 19####Consum
ed 20####Consumed 21####Consumed 22####Consumed 23####Produced 23####Produced 24####Produced 25####Produced 26####Produce
d 27####Produced 28####Produced 29####Produced 30####Produced 31####Produced 32####Produced 33####Consumed 24####Consumed
 25####Consumed 26####Consumed 27####Consumed 28####Consumed 29####Consumed 30####Consumed 31####Consumed 32####Consumed
33####Consumed 34####Produced 34####Produced 35####Produced 36####Produced 37####Produced 38####Produced 39####Produced 4
0####Produced 41####Produced 42####Produced 43####Produced 44####Consumed 35####Consumed 36####Consumed 37####Consumed 38
####Consumed 39####Consumed 40####Consumed 41####Consumed 42####Consumed 43####Consumed 44####Consumed 45####Produced 45#
###Produced 46####Produced 47####Produced 48####Produced 49####Produced 50####Produced 51####Produced 52####Produced 53##
##Produced 54####Produced 55####Consumed 46####Consumed 47####Consumed 48####Consumed 49####Consumed 50####Consumed 51###
#Consumed 52####Consumed 53####Consumed 54####Consumed 55####Consumed 56####Produced 56####Produced 57####Produced 58####
Produced 59####Produced 60####Produced 61####Produced 62####Produced 63####Produced 64####Produced 65####Produced 66####C
onsumed 57####Consumed 58####Consumed 59####Consumed 60####Consumed 61####Consumed 62####Consumed 63####Consumed 64####Co
nsumed 65####Consumed 66####Consumed 67####Produced 67####Consumed 68####Produced 68####Produced 69####Produced 70####Pro
duced 71####Produced 72####Produced 73####Produced 74####Produced 75####Produced 76####Produced 77####Produced 78####Cons
umed 69####Consumed 70####Consumed 71####Consumed 72####Consumed 73####Consumed 74####Consumed 75####Consumed 76####Consu
med 77####Consumed 78####Consumed 79####Produced 79####Produced 80####Produced 81####Produced 82####Produced 83####Produc
ed 84####Produced 85####Produced 86####Produced 87####Produced 88####Produced 89####Consumed 80####Consumed 81####Consume
d 82####Consumed 83####Consumed 84####Consumed 85####Consumed 86####Consumed 87####Consumed 88####Consumed 89####Consumed
 90####Produced 90####Produced 91####Produced 92####Produced 93####Produced 94####Produced 95####Produced 96####Produced
97####Produced 98####Produced 99####Produced 100####Consumed 91####Consumed 92####Consumed 93####Consumed 94####Consumed
95####Consumed 96####Consumed 97####Consumed 98####Consumed 99####Consumed 100##
--------------------------------
Process exited after 0.04616 seconds with return value 0
Press any key to continue . . .
```

Theory:

A thread is a path of execution within a process.

Multi-threading: achieve parallelism by dividing a process into multiple threads.

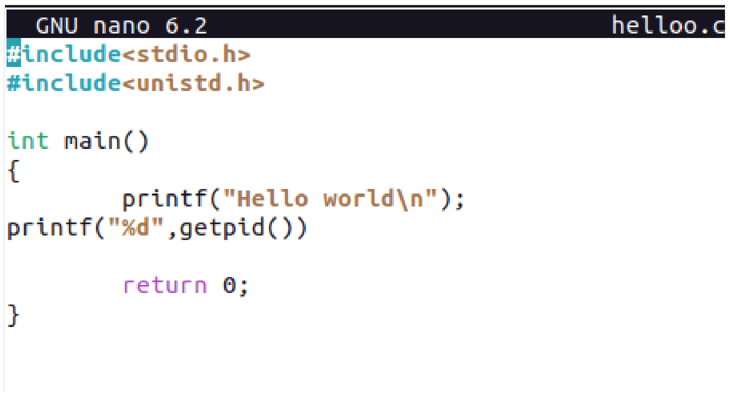
pthread_create: used to create a new thread

pthread_exit: used to terminate a thread.

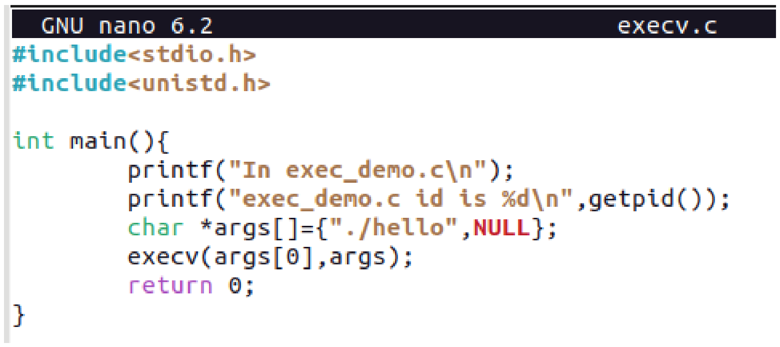
pthread_join: used to wait for the termination of a thread.

Problem:

Code:

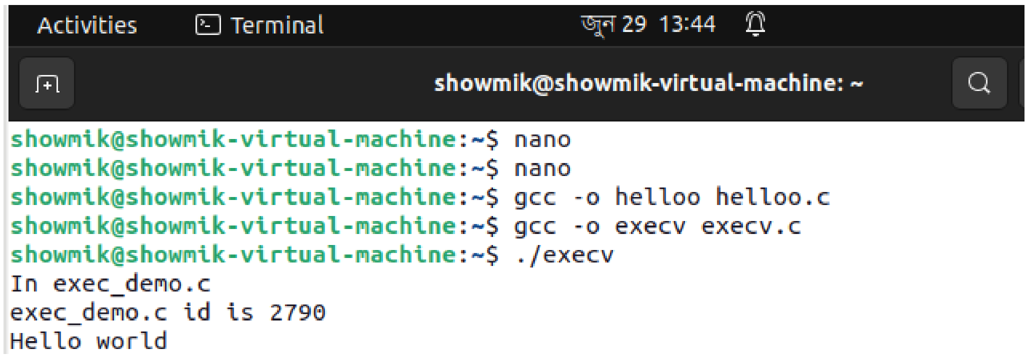


```
  GNU nano 6.2                              helloo.c
#include<stdio.h>
#include<unistd.h>

int main()
{
        printf("Hello world\n");
printf("%d",getpid())

        return 0;
}
```



```
  GNU nano 6.2                              execv.c
#include<stdio.h>
#include<unistd.h>

int main(){
        printf("In exec_demo.c\n");
        printf("exec_demo.c id is %d\n",getpid());
        char *args[]={"./hello",NULL};
        execv(args[0],args);
        return 0;
}
```

Output:



```
showmik@showmik-virtual-machine:~$ nano
showmik@showmik-virtual-machine:~$ nano
showmik@showmik-virtual-machine:~$ gcc -o helloo helloo.c
showmik@showmik-virtual-machine:~$ gcc -o execv execv.c
showmik@showmik-virtual-machine:~$ ./execv
In exec_demo.c
exec_demo.c id is 2790
Hello world
```