

ASSIGNMENT 2

MULTI THREADING

1. Write a pthread application where main task terminated but pending pthreads task still execute.

CODE :

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

pthread_t thread_id, thread_id2;

void *myThreadFun(void *vargp)
{
    printf("Printing from Thread 1\n");
    return NULL;
}

void *myThreadFun2(void *vargp)
{
    printf("Printing from Thread 2\n");
    return NULL;
}

int main()
{
    int ret = pthread_create(&thread_id, NULL, myThreadFun, NULL);
    int ret2 = pthread_create(&thread_id2, NULL, myThreadFun2, NULL);
    if (ret==0)
    {printf("All the thread are executing\n");}
```

```

pthread_exit(NULL);

printf("All the thread are not executing ");

return(0);
}

```

The screenshot shows a code editor with a C program and its execution output in a terminal window.

Code Editor:

```

1.c > main()
You, 2 seconds ago | 1 author (You)
1  /*1. Write a pthread application where main task terminated but pending pthreads task still
2  execute.*/
3  #include <stdio.h>...
7  pthread_t thread_id, thread_id2;
8  void *myThreadFun(void *vargp)
9  {
10     printf("Printing from Thread 1\n");
11     return NULL;
12 }
13 void *myThreadFun2(void *vargp)
14 {
15     printf("Printing from Thread 2 \n");
16     return NULL;
17 }
18 int main()
19 {
20     int ret = pthread_create(&thread_id, NULL, myThreadFun, NULL);
21     int ret2 = pthread_create(&thread_id2, NULL, myThreadFun2, NULL);
22     if (ret==0)
23     {printf("All the thread are executing\n");}
24     pthread_exit(NULL);
25     printf("All the thread are not executing ");
26     return(0);
27 }

```

Terminal Output:

```

om@om-VirtualBox:~/om/linux_internals/multithreding_assignment_2$ ./a.out
All the thread are executing
Printing from Thread 2
Printing from Thread 1
om@om-VirtualBox:~/om/linux_internals/multithreding_assignment_2$

```

2. Write a program where a structure of information is passed to pthread task function, and display structure of information.

CODE :

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

pthread_t thread_id;

struct multi{
    int s_value;
};

void *myThreadFun(void *arg){
    struct multi *value;
    value = (struct multi *) arg; // typecasting a structure
    printf("Printing from Thread 1\n");
    printf("value of struct is %d\n", value->s_value);
    return NULL;
}

int main()
{
    struct multi value1;
    value1.s_value=1000;
    int ret = pthread_create(&thread_id, NULL, myThreadFun, (void *)&value1);
    if (ret==0)
    {printf("Thread executing\n");}
```

```
pthread_join(thread_id,NULL);

return(0);

}
```

```
C 2.c > myThreadFun(void *)
You, 2 seconds ago | 1 author (You)
1 > /*2. Write a program where a structure of information passed to pthread task function, and-
3 > #include <stdio.h>-
7 pthread_t thread_id;
You, 2 days ago | 1 author (You)
8 struct multi{
9     int s_value;
10 };
11 void *myThreadFun(void *arg){
12     struct multi *value;
13     value = (struct multi *) arg; // typecasting a structure
14     printf("Printing from Thread 1\n");
15     printf("value of struct is %d\n", value->s_value);
16     return NULL;
17 }
18 int main()
19 {
20     struct multi value1;
21     value1.s_value=1000;
22     int ret = pthread_create(&thread_id, NULL, myThreadFun, (void *)&value1);
23     if (ret==0)
24     {printf("Thread executing\n");}
25     pthread_join(thread_id,NULL);
26     return(0);
27 }
28
29
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL GITLENS

```
om@om-VirtualBox:~/om/linux_internals/multithreding_assignment_2$ ./a.out
Thread executing
Printing from Thread 1
value of struct is 1000
om@om-VirtualBox:~/om/linux_internals/multithreding_assignment_2$
```

3. Write a pthread program that implements simple initialization code.

CODE:

```
#include <pthread.h>
```

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

```
pthread_once_t once = PTHREAD_ONCE_INIT;
```

```
void *myinit()
```

```
{  
    printf("\n in init function\n");  
}
```

```
void *mythread(void *i)
```

```
{  
    printf("\n i am in mythread:%d\n", (int *)i);  
    pthread_once(&once, (void *)myinit);  
    printf("\n exit from my thread:%d\n", (int *)i);  
}
```

```
int main()
```

```
{  
    pthread_t thread, thread1, thread2;  
    pthread_create(&thread, NULL, mythread, (void *)1);  
    pthread_create(&thread, NULL, mythread, (void *)2);  
    pthread_create(&thread, NULL, mythread, (void *)3);  
    printf("exiting main\n");  
    pthread_exit(0);  
}
```

}

```
C 3.c > main()
You, 2 days ago | 1 author (You)
1  #include <pthread.h>
2  #include <stdio.h>
3
4  pthread_once_t once = PTHREAD_ONCE_INIT;
5
6  void *myinit()
7  {
8      printf("\n in init function\n");
9  }
10
11 void *mythread(void *i)
12 {
13     printf("\n i am in mythread:%d\n", (int *)i);
14     pthread_once(&once, (void *)myinit);
15     printf("\n exit from my thread:%d\n", (int *)i);
16 }
17
18 int main()
19 {
20     pthread_t thread, thread1, thread2;
21     pthread_create(&thread, NULL, mythread, (void *)1);
22     pthread_create(&thread, NULL, mythread, (void *)2);
23     pthread_create(&thread, NULL, mythread, (void *)3);
24     printf("exiting main\n");
25     pthread_exit(0);
26 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL GITLENS

```
om@om-VirtualBox:~/om/linux_internals/multithreding_assignment_2$ ./a.out
exiting main

i am in mythread:3

in init function

exit from my thread:3

i am in mythread:2

exit from my thread:2

i am in mythread:1

exit from my thread:1
om@om-VirtualBox:~/om/linux_internals/multithreding_assignment_2$
```

4.write a program, which get and set pthread scheduling policy and priority.

CODE :

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

void *Proc(void *param)
{
    sleep(3);
    printf("inside thread function\n");
    return 0;
}

int main(int argc, char *argv[])
{
    pthread_attr_t Attr;
    pthread_t Id;
    int err, info;

    pthread_attr_init(&Attr);
    pthread_attr_getinheritsched(&Attr, &info);

    // before setting the privacy and policy(default)
    printf("value of info: %d\n", info); // 0 on inherit

    if (info == 0)
```

```
{  
    printf("INHERIT\n");  
}  
else  
{  
    printf("EXPLICIT\n");  
}  
  
// after setting the privacy and policy(modified)  
pthread_attr_setinheritsched(&Attr, PTHREAD_EXPLICIT_SCHED);  
pthread_attr_getinheritsched(&Attr, &info);  
  
printf("value of info: %d\n", info); // 1 on explicit  
  
if (info == 0)  
{  
    printf("INHERIT\n");  
}  
else  
{  
    printf("EXPLICIT\n");  
}  
  
pthread_create(&Id, &Attr, Proc, NULL);  
return 0;  
}
```



```
C 4.c > main(int, char * [])
16 pthread_attr_t Attr;
17 pthread_t Id;
18 int err, info;
19
20 pthread_attr_init(&Attr);
21 pthread_attr_getinheritsched(&Attr, &info);
22
23 // before setting the privacy and policy(default)
24 printf("value of info: %d\n", info); // 0 on inherit
25
26 if (info == 0)
27 {
28     printf("INHERIT\n");
29 }
30 else
31 {
32     printf("EXPLICIT\n");
33 }
34
35 // after setting the privacy and policy(modified)
36 pthread_attr_setinheritsched(&Attr, PTHREAD_EXPLICIT_SCHED);
37 pthread_attr_getinheritsched(&Attr, &info);
38
39 printf("value of info: %d\n", info); // 1 on explicit
40
41 if (info == 0)
42 {
    You, 2 days ago • adding assignment ...
}

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL GITLENS

om@om-VirtualBox:~/om/linux_internals/multithreding_assignment_2$ ./a.out
value of info: 0
INHERIT
value of info: 1
EXPLICIT
om@om-VirtualBox:~/om/linux_internals/multithreding_assignment_2$
```

5. Write a program that implements threads synchronization using pthread spinlock techniques.

CODE :

```
#include <pthread.h>
```

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

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

```
#include <semaphore.h>
```

```
static pthread_spinlock_t spinlock;
```

```
volatile int lock_s; // using volatile we can change the lock_s value anytime inbetween
```

```
void *spinLockThread(void *i)
```

```
{
```

```
    int rc;
```

```
    int count = 0;
```

```
    printf("entered thread %d, getting spin lock\n", (int *)i);
```

```
    printf("%d thread complete\n", (int *)i);
```

```
    return NULL;
```

```
}
```

```
int main()
```

```
{
```

```
    int rc = 0;
```

```
    pthread_t thread;
```

```
if (pthread_spin_init(&lock_s, PTHREAD_PROCESS_PRIVATE) != 0)
{
    perror("init");
}
```

```
printf("getting the spin lock\n");
rc = pthread_spin_lock(&lock_s);
```

```
printf("creating the spin lock\n");
```

```
rc = pthread_create(&thread, NULL, spinLockThread, (int *)1);
```

```
printf("hold the lock for 5 secs...\n");
sleep(5);
```

```
printf("unlocking the lock\n");
rc = pthread_spin_unlock(&lock_s);
```

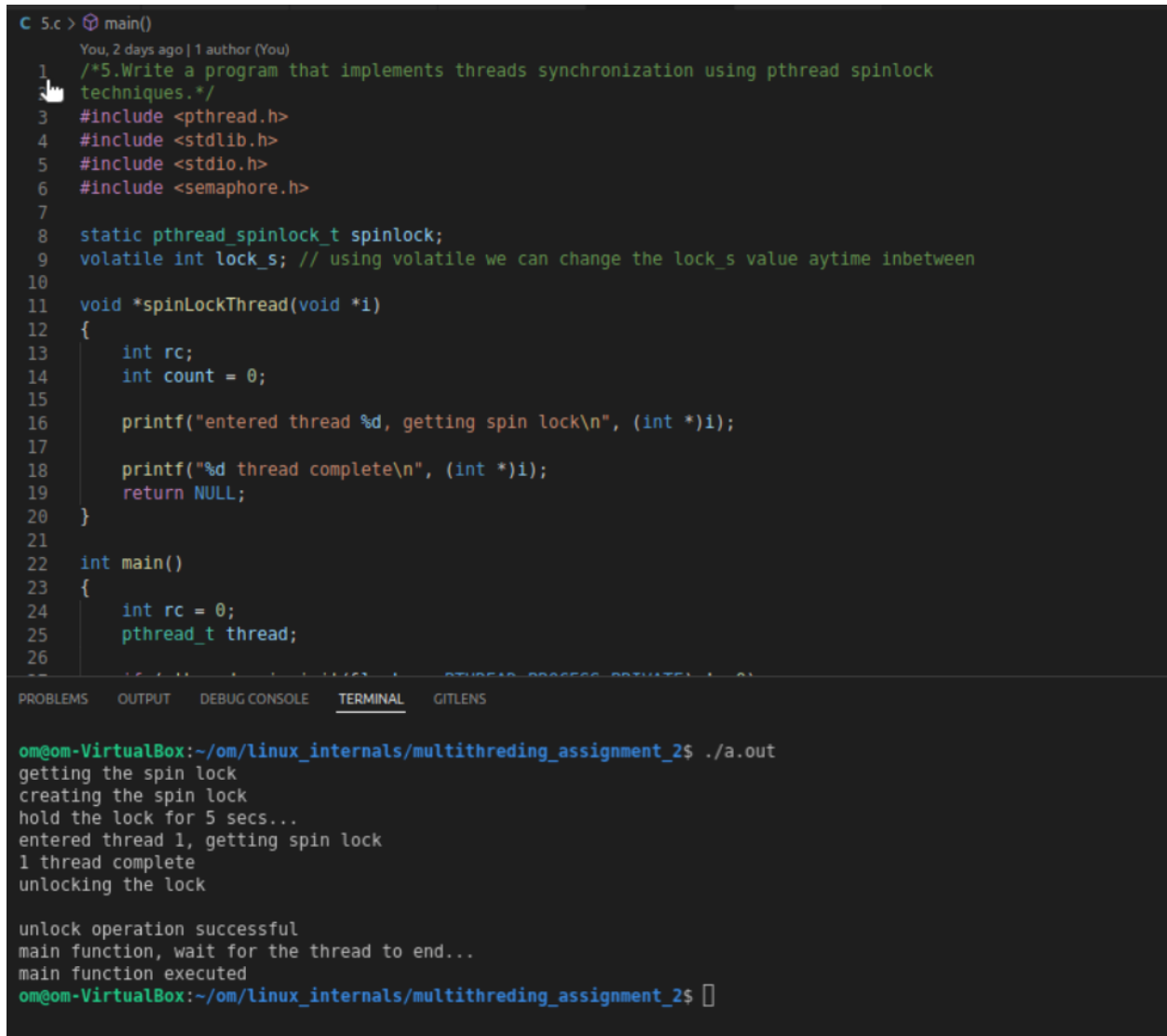
```
if (rc == 0)
{
    printf("\nunlock operation successful\n");
}
else
{
    printf("\nunlock operation unsuccessful\n");
}
```

```
printf("main function, wait for the thread to end...\n");

rc = pthread_join(thread, NULL);

printf("main function executed\n");

return 0;
}
```



The image shows a code editor with a C program and its terminal output. The C program implements a spinlock using pthread. It includes headers for pthread, stdlib, stdio, and semaphore. A static pthread_spinlock_t variable is declared, and a volatile int lock_s is used to manage the lock state. A spinLockThread function is defined, which prints messages when it enters and leaves the lock. The main function creates the thread, calls pthread_join to wait for it, and prints a message when the thread has finished.

```
C 5.c > main()
You, 2 days ago | 1 author (You)
1  /*5.Write a program that implements threads synchronization using pthread spinlock
   techniques.*/
3  #include <pthread.h>
4  #include <stdlib.h>
5  #include <stdio.h>
6  #include <semaphore.h>
7
8  static pthread_spinlock_t spinlock;
9  volatile int lock_s; // using volatile we can change the lock_s value anytime inbetween
10
11 void *spinLockThread(void *i)
12 {
13     int rc;
14     int count = 0;
15
16     printf("entered thread %d, getting spin lock\n", (int *)i);
17
18     printf("%d thread complete\n", (int *)i);
19     return NULL;
20 }
21
22 int main()
23 {
24     int rc = 0;
25     pthread_t thread;
26
27     // ... (rest of the code is obscured by a horizontal line) ...

```

Below the code editor, the terminal output is shown:

```
om@om-VirtualBox:~/om/linux_internals/multithreding_assignment_2$ ./a.out
getting the spin lock
creating the spin lock
hold the lock for 5 secs...
entered thread 1, getting spin lock
1 thread complete
unlocking the lock

unlock operation successful
main function, wait for the thread to end...
main function executed
om@om-VirtualBox:~/om/linux_internals/multithreding_assignment_2$
```