Trainee name : Khushi Mordani
Employee id : 150111
Linux Internals – Multithreading Assignment

1.Write a multithreading program, where threads runs same shared golbal variable of the process between them.

Source code:

```
//Khushi Mordani
/*1.Write a multithreading program, where threads runs same shared global variable of the process between them.*/
#include<stdio.h>
#include<stdlib.h>
#include<pthread.h>

int sharedvar=49;

void *thread_mul(void *arg)
{
        int n;
        printf("Enter a number to multiply:");
        scanf("%d",&n);
        sharedvar = sharedvar * n ;
        printf("After multiplication,shared variable is %d\n",sharedvar);
}

void *thread_sub(void *arg)
{
        int i;
        printf("Enter a number to subtract:");
        scanf("%d",&i);
        sharedvar = sharedvar - i;
        printf("\nAfter subtracting,shared variable is %d\n",sharedvar);
}

int main()
{
        pthread_t thread1,thread2;
        pthread_create(&thread1,NULL,thread_mul,NULL);
        pthread_create(&thread2,NULL,thread_sub,NULL);
        pthread_join(thread1,NULL);
        pthread_join(thread2,NULL);
        printf("Shared variable final value= %d\n",sharedvar);
        return 0;
}
```

Output:

```
khushi@khushi-VirtualBox:~$ gcc L1A3Q1.c -lpthread
khushi@khushi-VirtualBox:~$ ./a.out
Enter a number to subtract:8
Enter a number to multiply:
After subtracting,shared variable is 41
2
After multiplication,shared variable is 82
Shared variable final value= 82
```

2.Write a program where thread cancel itself.(use pthread_cancel())

Source code:

```
//Khushi Mordani
//2.Write a program where thread cancel itself.(use pthread_cancel())
#include<pthread.h>
#include<stdio.h>
#include<unistd.h>

void *hello(void *threadid)
{
        printf("Hello,I am Khushi!!\n");
        while(1);{}
}

int main()
{
        pthread_t t1;
        int rc, t=0;
        printf("Creating thread\n");
        rc = pthread_create(&t1,NULL,hello,NULL);
        printf("thread id: %lu\n",t1);
        sleep(5);
        rc = pthread_cancel(t1);
        if(t==0)
        {
                printf("Thread cancelled!\n");
                printf("thread id: %lu\n",t1);
        }
        return 0;
}
```

Output:

```
khushi@khushi-VirtualBox:~$ gcc L1A3Q2.c -lpthread
khushi@khushi-VirtualBox:~$ ./a.out
Creating thread
thread id: 140301779785472
Hello,I am Khushi!!
Thread cancelled!
thread id: 140301779785472
```

3. Write a program that changes the default properties of newly created posix threads.(ex: to change default pthread stack size)

Source code:

```
//Khushi Mordani
//3. Write a program that changes the default properties of newly created posix threads.(ex: to change default pthread stack size )
#include<stdio.h>
#include<pthread.h>
#include<unistd.h>
#include<stdlib.h>
#include<sys/mman.h>

void* change(void* par)
{
        sleep(2);
        return 0;
}

int main()
{
        pthread_attr_t Attr;
        pthread_t id;
        void *stack;
        size_t siz;
        int err;
        size_t my_stacksize = 0x30000000;
        void * my_stack;

        pthread_attr_init(&Attr);

        pthread_attr_getstacksize(&Attr, &siz);
        pthread_attr_getstackaddr(&Attr, &stack);

        printf("Default address: %08x Default size: %d\n",stack,siz);

        my_stack = (void*)malloc(my_stacksize);

        pthread_attr_setstack(&Attr,my_stack,my_stacksize);
        pthread_create(&id,&Attr,change,NULL);
        pthread_attr_getstack(&Attr,&stack,&siz);
        printf("Newly derived stack address:%08x and size:%d\n",stack,siz);
        sleep(2);
        exit(0);
}
```

Output:

```
khushi@khushi-VirtualBox:~$ gcc L1A3Q3.c -lpthread
L1A3Q3.c: In function ‘main’:
L1A3Q3.c:28:2: warning: ‘pthread_attr_getstackaddr’ is deprecated [-Wdeprecated-declarations]
  pthread_attr_getstackaddr(&Attr, &stack);
  ^~~~~~~~~~~~~~~~~~~~~~~~
In file included from L1A3Q3.c:4:0:
/usr/include/pthread.h:356:12: note: declared here
 extern int pthread_attr_getstackaddr (const pthread_attr_t *__restrict
            ^~~~~~~~~~~~~~~~~~~~~~~~
L1A3Q3.c:30:30: warning: format ‘%x’ expects argument of type ‘unsigned int’, but argument 2 has type ‘void *’ [-Wformat=]
  printf("Default address: %08x Default size: %d\n",stack,siz);
                           ~~~^
                           %08p
L1A3Q3.c:30:47: warning: format ‘%d’ expects argument of type ‘int’, but argument 3 has type ‘size_t {aka long unsigned int}’ [-Wformat=]
  printf("Default address: %08x Default size: %d\n",stack,siz);
                                              ~^
                                              %ld
L1A3Q3.c:37:41: warning: format ‘%x’ expects argument of type ‘unsigned int’, but argument 2 has type ‘void *’ [-Wformat=]
  printf("Newly derived stack address:%08x and size:%d\n",stack,siz);
                                      ~~~^
                                      %08p
L1A3Q3.c:37:53: warning: format ‘%d’ expects argument of type ‘int’, but argument 3 has type ‘size_t {aka long unsigned int}’ [-Wformat=]
  printf("Newly derived stack address:%08x and size:%d\n",stack,siz);
                                                     ~^
                                                     %ld
/tmp/ccLEwg7E.o: In function `main':
L1A3Q3.c:(.text+0x6a): warning: the use of `pthread_attr_getstackaddr' is deprecated, use `pthread_attr_getstack'
khushi@khushi-VirtualBox:~$ ./a.out
Default address: 00000000 Default size: 8388608
Newly derived stack address:43004010 and size:805306368
```

4. Write a program where pthread task displays the thread id and also prints the calling process pid.

Source code:

```
//Khushi Mordani
//4. Write a program where pthread task displays the thread id and also prints the calling process pid.

#include<pthread.h>
#include<stdio.h>
pthread_t thread_id;
void *thread_func(void *arg)
{
        pid_t pid;
        pthread_t thread_id;
        pid=getpid();
        thread_id=pthread_self();
        printf("Process id: %u Thread id:%u \n",(unsigned int)pid,(unsigned int)thread_id);
        return 0;
}

int main(void)
{
        int create;
        create=pthread_create(&thread_id,NULL,thread_func,NULL);
        if(create!=0)
        printf("Cannot create thread: %s\n",strerror(create));
        while(1);
        exit(0);
}
```

Output:

```
khushi@khushi-VirtualBox:~$ gcc L1A3Q4.c -lpthread
L1A3Q4.c: In function ‘thread_func’:
L1A3Q4.c:11:6: warning: implicit declaration of function ‘getpid’; did you mean ‘fgetpos’? [-Wimplicit-function-declaration]
  pid=getpid();
      ^~~~~~
      fgetpos
L1A3Q4.c: In function ‘main’:
L1A3Q4.c:22:38: warning: implicit declaration of function ‘strerror’; did you mean ‘perror’? [-Wimplicit-function-declaration]
  printf("Cannot create thread: %s\n",strerror(create));
                                     ^~~~~~~~
                                     perror
L1A3Q4.c:22:33: warning: format ‘%s’ expects argument of type ‘char *’, but argument 2 has type ‘int’ [-Wformat=]
  printf("Cannot create thread: %s\n",strerror(create));
                               ~^    ~~~~~~~~~~~~~~~~
                               %d
L1A3Q4.c:24:2: warning: implicit declaration of function ‘exit’ [-Wimplicit-function-declaration]
  exit(0);
  ^~~~
L1A3Q4.c:24:2: warning: incompatible implicit declaration of built-in function ‘exit’
L1A3Q4.c:24:2: note: include ‘<stdlib.h>’ or provide a declaration of ‘exit’
khushi@khushi-VirtualBox:~$ ./a.out
Process id: 4274 Thread id:58119936
```

5.Write a program that implements threads synchronization using mutex techniques.

Source code:

```
//Khushi Mordani
//5.Write a program that implements threads synchronization using mutex techniques.

#include<stdio.h>
#include<stdlib.h>
#include<pthread.h>
#include<semaphore.h>

int sharedvar=87;

pthread_mutex_t my_mutex; //create mutex
void *thread_add(void *arg)
{
        pthread_mutex_lock(&my_mutex); //take mutex
        int n;
        printf("Enter a number to add:\n");
        scanf("%d",&n);
        sharedvar=sharedvar+n;
        printf("After addition,shared variable is %d\n",sharedvar);
        pthread_mutex_unlock(&my_mutex);        //release mutex
}

void *thread_sub(void *arg)
{
        pthread_mutex_lock(&my_mutex);  //take mutex
        int i;
        printf("Enter a number to subtract:");
        scanf("%d",&i);
        sharedvar=sharedvar-i;
        printf("After subtraction,shared variable is %d\n",sharedvar);
        pthread_mutex_unlock(&my_mutex);        //release semaphore
}

int main()
{
```

```
        pthread_t thread1,thread2;
        pthread_mutex_init(&my_mutex,NULL);
        pthread_create(&thread1,NULL,thread_add,NULL);
        pthread_create(&thread2,NULL,thread_sub,NULL);
        pthread_join(thread1,NULL);
        pthread_join(thread2,NULL);
        printf("Shared variable final value = %d\n",sharedvar);
        return 0;
}
```

Output: