BITS, Pilani - Hyderabad Campus
Operating Systems (CS F372)

Tutorial – 6

You will learn about Threads and Linux process scheduling in this tutorial.

Threads

Threads are a mechanism to implement parallelism that allows one to spawn a new concurrent process flow. It is most effective on multi-processor systems where the process flow can be scheduled to run on multiple processors thus gaining speed through parallel processing.

The following program *mythread.c* attempts to use threads to increment *sum* to 5000:

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

#define N 5
int sum = 0;
void* fun () {
  int i;
  for (i = 0; i < 1000; i++ ) {
      sum += 1;
  }
  return NULL;
}

int main (int argc, char* argv[])
{
  pthread_t t[N];
  int data[N], errcode, i;
  for (i = 0; i < N; i ++) {
      data [i] = atoi (argv[i + 1]);
      if (pthread_create (&t[i], NULL, fun, NULL)) {
          printf ("Error creating thread\n");
          exit(-1);
      }
  }
  for (i = 0; i < N; i ++) {
      pthread_join (t[i], NULL);
  }
  printf ("Sum: %d\n", sum);
}
```

Compilation of a program with threads:

```
$ cc -pthread  <yourfilename.c>   -o  <yourexecname>
```

Note the –pthread option to cc

What is the expected output? Run it multiple times. Explain the behaviour. Can you think of anything to fix it?

Understand the following basic functions used in thread programming:

pthread_create :

```
int pthread_create(     pthread_t * thread, const pthread_attr_t * attr,
                        void * (*start_routine)(void *), void *arg);
```

You will learn about the 2nd parameter *attr* in a later tutorial; it basically sets attributes of the new thread.

pthread_join :

How is synchronization achieved with pthread_join? Repeat the Hello World program without pthread_join and observe the difference.

```
int pthread_join(pthread_t tid, void **thread_return);
```

pthread_exit :

```
void pthread_exit(void *retval);
```

pthread_self :

```
pthread_t pthread_self(void);
```

Scheduling in Linux:

The latest Linux kernel supports the following scheduling policies. SCHED_OTHER/SCHED_NORMAL is the default policy on the desktop that use CFS scheduler. Linux supports the following policies :

SCHED_FIFO : Real-time scheduling policy that uses the First In-First Out scheduling algorithm.
SCHED_RR: Real-time scheduling policy that uses the Round Robin scheduling algorithm.
SCHED_OTHER: Default Linux scheduling policy used for regular tasks.
SCHED_BATCH : Scheduling policy designed for CPU-intensive tasks. It does not preempt nearly as often as regular tasks would, thereby allowing tasks to run longer but at the cost of interactivity. This is well suited for batch jobs.
SCHED_IDLE: Scheduling policy intended for *very* low prioritized tasks.

You can launch a programe with your required priority using *nice*

```
$ nice -n nice_value program_name
```

You can also change the priority of an already running process using *renice*

```
$ renice -n nice_value -p process_id
```

The **chrt** command sets or retrieves the real-time scheduling attributes of a running process, or runs a command with the specified attributes. You can get or retrieve both the scheduling policy and priority (if applicable) of a process. In the following example, a bash process whose PID is 1784 is used. To *retrieve* the real-time attributes of an existing task:

```
sss@sss-lubuntu:~$ chrt -p 1784
pid 1784's current scheduling policy: SCHED_OTHER
pid 1784's current scheduling priority: 0
```

Before setting a new scheduling policy on the process, you need to find out the minimum and maximum valid priorities for each scheduling algorithm:

```
sss@sss-lubuntu:~$ chrt -m
SCHED_OTHER min/max priority  : 0/0
SCHED_FIFO min/max priority   : 1/99
SCHED_RR min/max priority     : 1/99
SCHED_BATCH min/max priority  : 0/0
SCHED_IDLE min/max priority   : 0/0
```

In the above example, SCHED_OTHER, SCHED_BATCH, SCHED_IDLE polices only allow for static priority 0, while that of SCHED_FIFO and SCHED_RR can range from 1 to 99.
So you only have the default static priority 0 in the desktop (`SCHED_OTHER`). Remember that the kernel will internally modify the priority of a process called dynamic priority.

Nice values can affect the weight of the running process's priority. So, for two processes with the same priority, different nice values would give different weight and hence different CPU share to the processes

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

int main(){
        int p = getpid();
        int pri = getpriority(PRIO_PROCESS, p);
        printf("The original priority of %d is :%d\n",p, pri);
        setpriority(PRIO_PROCESS,p, 15);

        int pp = getpriority(PRIO_PROCESS, p);
        printf("priority is set to :%d\n", pp);
       while(1){}
}
```

Homework exercises:

1. Write a program that creates two threads that alternately print their thread ID and consecutive numbers from 1 to 20. After printing, a thread should sleep for one second.

   `$ ./threadprogram`

   ```
   Thread 1023 printing 1
   Thread 1024 printing 2
   Thread 1023 printing 3
   Thread 1024 printing 4
   ...
   Thread 1023 printing 19
   Thread 1024 printing 20
   ```

2. Find the problem with the following piece of code that was supposed to print numbers from 0 to 9.

   ```
   #include <pthread.h>
   #include <stdio.h>
   void* fun(void* ptr) {
       int i = *((int *) ptr);
       printf("%d ", i);
       return NULL;
   }
   int main() {
            int i;
            pthread_t tid[10];
            for(i =0; i < 10; i++) {
               pthread_create(&tid[i], NULL, fun, &i);
            }
            pthread_exit(NULL);
   }
   ```

3. Write a C program to change the scheduling policy of a process via system calls (no system() command) to `SCHED_FIFO` and the priority to 80. The process whose priority you are changing should be a background process. e.g. you can create one by `$ sleep 1000&`
   Find information on the following from man pages; you may need them:
   `struct sched_param, sched_priority(), sched_setscheduler()`