OPERATING SYSTEMS

Threads Creation and Thread Scheduling

Nitin V Pujari
Faculty, Computer Science
Dean - IQAC, PES University

OPERATING SYSTEMS

Course Syllabus - Unit 2

12 Hours

Unit 2: Threads & Concurrency

Introduction to Threads, types of threads, Multicore Programming, Multithreading Models, Thread creation, Thread Scheduling, PThreads and Windows Threads, Mutual Exclusion and Synchronization: software approaches, principles of concurrency, hardware support, Mutex Locks, Semaphores. Classic problems of Synchronization: Bounded-Buffer Problem, Readers -Writers problem, Dining Philosophers Problem concepts. Synchronization Examples - Synchronisation mechanisms provided by Linux/Windows/Pthreads. Deadlocks: principles of deadlock, tools for detection and Prevention.

OPERATING SYSTEMS

Course Outline

13	Introduction to Threads, types of threads, Multicore Programming, Multithreading Models	4.1 - 4.3	42.8
14	Thread creation, Thread Scheduling	5.4	
15	Pthreads and Windows Threads	4.4	
16	Mutual Exclusion and Synchronization: software approaches,	6.1-6.2	
17	principles of concurrency, hardware support	6.3-6.4	
18	Mutex Locks, Semaphores	6.5, 6.6	
19	Classic problems of Synchronization: Bounded-Buffer Problem, Readers-Writers problem	6.7-6.8	
20	Dining-Philosophers Problem	6.8	
21	Synchronization Examples: Synchronisation mechanisms provided by Linux/Windows/Pthreads.	6.9	
22	Deadlocks: principles of deadlock, Deadlock Characterization	7.1-7.3	
23	Deadlock Prevention, Deadlock example	7.4-7.5	
24	Deadlock Detection, Algorithm	7.6	

Topic Outline

- Threads and Concurrency
    - Thread Scheduling
    - PThread Example

Thread Scheduling

- Distinction between user-level and kernel-level threads
- When threads supported, threads are scheduled, not processes
- Many-to-one and many-to-many models, thread library schedules user-level threads to run on LWP
  - Known as Process-Contention Scope (PCS) since scheduling competition is within the process
  - Typically done via priority set by programmer
- Kernel thread scheduled onto available CPU is known as System Contention Scope (SCS) – competition among all threads in system

PThread Scheduling

- API allows specifying either PCS or SCS during thread creation
- PTHREAD_SCOPE_PROCESS schedules threads using PCS scheduling
- PTHREAD_SCOPE_SYSTEM schedules threads using SCS scheduling
- Can be limited by OS_Linux and Mac OS X only allow PTHREAD_SCOPE_SYSTEM

Thread Creation

- Process creation is heavy-weight while thread creation is light-weight
- Most modern applications are multithreaded
- Threads run within application
- Multiple tasks in application can be implemented by threads
  - Update display
  - Fetch data
  - Spell checking
- Can simplify code, increase efficiency
- Kernels are generally multithreaded

Thread Creation - pthreads

OPERATING SYSTEMS

Thread Creation - pthread example

```c
// C program Maximum Threads in a processs
// On Success, pthread_create returns 0 and
// on Error, it returns the error number
// Program has to be compilied using
// gcc Threads_Max.c -pthread
// Because pthread has to external linked else results linker error

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


void *Thread ( void *TParameter)
{

}

int main()
{
    int TCreationError = 0, ThreadCount = 0;
    pthread_t ThreadId;

    while (TCreationError == 0)
    {
        TCreationError = pthread_create (&ThreadId, NULL, Thread, NULL);
        printf("\nThreadCount=>%d, ThreadId=>%d",ThreadCount, ThreadId);
        ThreadCount++;
    }

    printf("\nMaximum threads that can be created within a Process in this system"
           " is : %d", ThreadCount);
}
```

OPERATING SYSTEMS

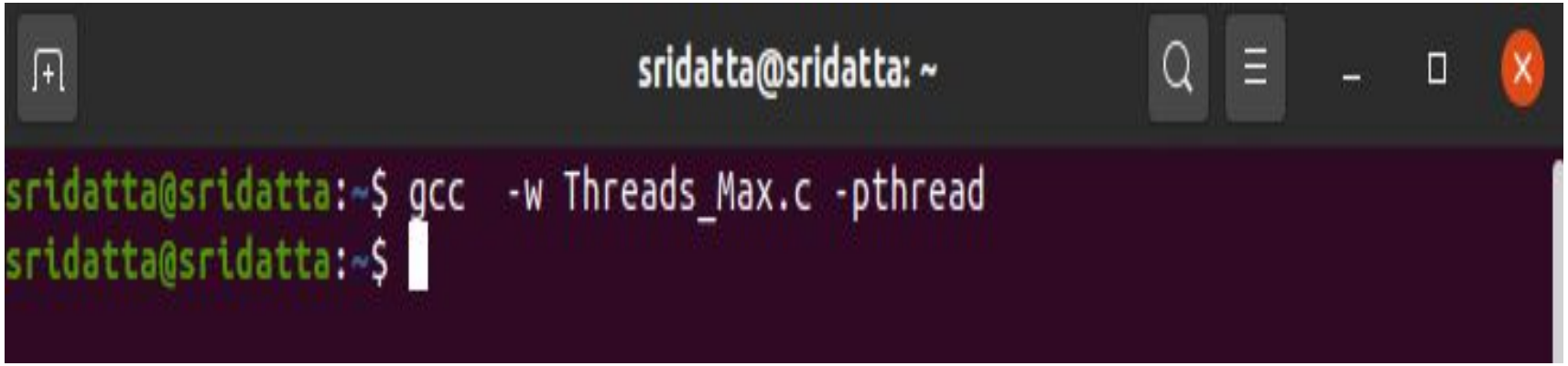
Thread Creation - pthread example

```
sridatta@sridatta:~$ gcc  Threads_Max.c
Threads_Max.c: In function ‘main’:
Threads_Max.c:21:47: warning: format ‘%d’ expects argument of type ‘int’, but ar
gument 3 has type ‘pthread_t’ {aka ‘long unsigned int’} [-Wformat=]
   21 |         printf("\nThreadCount=>%d, ThreadId=>%d",ThreadCount, ThreadId);
      |                                              ~^                ~~~~~~~~
      |                                               |                |
      |                                               int              pthread_t
      |
{aka long unsigned int}
      |                                              %ld
/usr/bin/ld: /tmp/ccP7xXd5.o: in function `main':
Threads_Max.c:(.text+0x53): undefined reference to `pthread_create'
collect2: error: ld returned 1 exit status
sridatta@sridatta:~$
```

Thread Creation - pthread example

```
sridatta@sridatta:~$ gcc  -w Threads_Max.c
/usr/bin/ld: /tmp/ccmPPuCa.o: in function `main':
Threads_Max.c:(.text+0x53): undefined reference to `pthread_create'
collect2: error: ld returned 1 exit status
sridatta@sridatta:~$
```

Thread Creation - pthread example



OPERATING SYSTEMS

Thread Creation - pthread example

```
ThreadCount=>32702, ThreadId=>1097725696
ThreadCount=>32703, ThreadId=>1089332992
ThreadCount=>32704, ThreadId=>1080940288
ThreadCount=>32705, ThreadId=>1072547584
ThreadCount=>32706, ThreadId=>1064154880
ThreadCount=>32707, ThreadId=>1055762176
ThreadCount=>32708, ThreadId=>1047369472
ThreadCount=>32709, ThreadId=>1038976768
ThreadCount=>32710, ThreadId=>1030584064
ThreadCount=>32711, ThreadId=>1022191360
ThreadCount=>32712, ThreadId=>1013798656
ThreadCount=>32713, ThreadId=>1005405952
ThreadCount=>32714, ThreadId=>997013248
ThreadCount=>32715, ThreadId=>988620544
ThreadCount=>32716, ThreadId=>980227840
ThreadCount=>32717, ThreadId=>971835136
ThreadCount=>32718, ThreadId=>963442432
ThreadCount=>32719, ThreadId=>955049728
ThreadCount=>32720, ThreadId=>946657024
ThreadCount=>32721, ThreadId=>938264320
ThreadCount=>32722, ThreadId=>929871616
ThreadCount=>32723, ThreadId=>921478912
ThreadCount=>32724, ThreadId=>913086208
ThreadCount=>32725, ThreadId=>904693504
ThreadCount=>32726, ThreadId=>896300800
ThreadCount=>32727, ThreadId=>887908096
ThreadCount=>32728, ThreadId=>879515392
ThreadCount=>32729, ThreadId=>871122688
ThreadCount=>32730, ThreadId=>862729984
ThreadCount=>32731, ThreadId=>854337280
ThreadCount=>32732, ThreadId=>845944576
ThreadCount=>32733, ThreadId=>837551872
ThreadCount=>32734, ThreadId=>829159168
ThreadCount=>32735, ThreadId=>820766464
ThreadCount=>32736, ThreadId=>812373760
ThreadCount=>32737, ThreadId=>803981056
ThreadCount=>32738, ThreadId=>795588352
ThreadCount=>32739, ThreadId=>787195648
ThreadCount=>32740, ThreadId=>778802944
ThreadCount=>32741, ThreadId=>770410240
ThreadCount=>32742, ThreadId=>762017536
ThreadCount=>32743, ThreadId=>753624832
ThreadCount=>32744, ThreadId=>745232128
ThreadCount=>32745, ThreadId=>736839424
ThreadCount=>32746, ThreadId=>728446720
ThreadCount=>32747, ThreadId=>720054016
ThreadCount=>32748, ThreadId=>711661312
ThreadCount=>32749, ThreadId=>703268608
ThreadCount=>32750, ThreadId=>694875904
ThreadCount=>32751, ThreadId=>694875904
Maximum threads that can be created within a Process in this system is : 32752
```

OPERATING SYSTEMS

Thread Scheduling - pthread example

```c
// C program Maximum Threads in a processs
// On Success, pthread_create returns 0 and
// on Error, it returns the error number
// Program has to be compilied using
// gcc Threads_Scheduling.c -pthread
// Because pthread has to be externally linked else results linker error

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

#define Threads_Max 5


void *Thread ( void *TParameter)
{

}

int main(int argc, char **argv)
{
    int i;
    int ThreadScope;

    pthread_t Threads[Threads_Max];
    pthread_attr_t ThreadAttributes;

    pthread_attr_init(&ThreadAttributes);

    ThreadScope = pthread_attr_getscope(&ThreadAttributes,&ThreadScope);
    printf("\n The current Thread Scope=>%d",ThreadScope);



}
```

```
sridatta@sridatta:~$ gcc  -w Threads_Scheduling.c -pthread
sridatta@sridatta:~$ ./a.out

The current Thread Scope=>0sridatta@sridatta:~$
```

OPERATING SYSTEMS

Thread Scheduling - pthread example

```c
// C program Maximum Threads in a processs
// On Success, pthread_create returns 0 and
// on Error, it returns the error number
// Program has to be compilied using
// gcc Threads_Scheduling.c -pthread
// Because pthread has to be externally linked else results linker error

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

#define Threads_Max 5


void *Thread ( void *TParameter)
{

}

int main(int argc, char **argv)
{
    int i;
    int ThreadScope;

    pthread_t Threads[Threads_Max];
    pthread_attr_t ThreadAttributes;

    pthread_attr_init(&ThreadAttributes);

    ThreadScope = pthread_attr_getscope(&ThreadAttributes,&ThreadScope);
    printf("\n The current Thread Scope=>%d",ThreadScope);
    printf("\n The Current PTHREAD_SCOPE_PROCESS=>%d",PTHREAD_SCOPE_PROCESS);
    printf("\n The Current PTHREAD_SCOPE_SYSTEM=>%d",PTHREAD_SCOPE_SYSTEM);



}
```

```
sridatta@sridatta:~$ gcc  -w Threads_Scheduling.c -pthread
sridatta@sridatta:~$ ./a.out

 The current Thread Scope=>0
 The Current PTHREAD_SCOPE_PROCESS=>1
 The Current PTHREAD_SCOPE_SYSTEM=>0sridatta@sridatta:~$
```

OPERATING SYSTEMS

Thread Scheduling - pthread example

```c
// C program Maximum Threads in a processs
// On Success, pthread_create returns 0 and
// on Error, it returns the error number
// Program has to be compilied using
// gcc Threads_Scheduling.c -pthread
// Because pthread has to be externally linked else results linker error

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

#define Threads_Max 5


void *Thread ( void *TParameter)
{

}

int main(int argc, char **argv)
{
    int i;
    int ThreadScope;

    pthread_t Threads[Threads_Max];
    pthread_attr_t ThreadAttributes;

    pthread_attr_init(&ThreadAttributes);

    ThreadScope = pthread_attr_getscope(&ThreadAttributes,&ThreadScope);
    printf("\n The current Thread Scope=>%d",ThreadScope);
    printf("\n The Current PTHREAD_SCOPE_PROCESS=>%d",PTHREAD_SCOPE_PROCESS);
    printf("\n The Current PTHREAD_SCOPE_SYSTEM=>%d",PTHREAD_SCOPE_SYSTEM);

    if(ThreadScope!=0)
        fprintf(stderr,"\nUnable to get scheduling scope information");
    else
        fprintf(stdout,"\n Got it");


    printf("\n\n\n\n");

}
```

```
sridatta@sridatta:~$ gcc  -w Threads_Scheduling.c -pthread
sridatta@sridatta:~$ ./a.out

 The current Thread Scope=>0
 The Current PTHREAD_SCOPE_PROCESS=>1
 The Current PTHREAD_SCOPE_SYSTEM=>0
 Got it



sridatta@sridatta:~$
```

OPERATING SYSTEMS

Thread Scheduling - pthread example

```c
// C program Maximum Threads in a processs
// On Success, pthread_create returns 0 and
// on Error, it returns the error number
// Program has to be compilied using
// gcc Threads_Scheduling.c -pthread
// Because pthread has to be externally linked else results linker error

// Linux supports PTHREAD_SCOPE_SYSTEM, but not PTHREAD_SCOPE_PROCESS.


#include<stdio.h>
#include<pthread.h>
#define Threads_Max 5


void *Thread ( void *TParameter)
{

}

int main(int argc, char **argv)
{
    int i;
    int ThreadScope;

    pthread_t Threads[Threads_Max];
    pthread_attr_t ThreadAttributes;

    pthread_attr_init(&ThreadAttributes);
    ThreadScope = pthread_attr_getscope(&ThreadAttributes,&ThreadScope);
    printf("\n The current Thread Scope=>%d",ThreadScope);
    printf("\n The Current PTHREAD_SCOPE_PROCESS=>%d",PTHREAD_SCOPE_PROCESS);
    printf("\n The Current PTHREAD_SCOPE_SYSTEM=>%d",PTHREAD_SCOPE_SYSTEM);

    if(ThreadScope!=0)
        fprintf(stderr,"\nUnable to get scheduling scope information");

    if(ThreadScope==PTHREAD_SCOPE_PROCESS)
        fprintf(stdout,"\n Currently the Scope is=> Process Contention Scope");
    else
        fprintf(stdout,"\n Currently the Scope is=> System Contention Scope");

    ThreadScope = 1;
    pthread_attr_setscope(&ThreadAttributes,PTHREAD_SCOPE_PROCESS);
    ThreadScope = pthread_attr_getscope(&ThreadAttributes,&ThreadScope);
    printf("\n The current Thread Scope after Change=>%d",ThreadScope);
    printf("\n\n\n\n");
}
```

```
pg@chairperson-HP:~$ gcc Threads_Scheduling.c -pthread
pg@chairperson-HP:~$ ./a.out

 The current Thread Scope=>0
 The Current PTHREAD_SCOPE_PROCESS=>1
 The Current PTHREAD_SCOPE_SYSTEM=>0
 Currently the Scope is=> System Contention Scope
 The current Thread Scope after Change=>0



pg@chairperson-HP:~$ gcc Threads_Scheduling.c -pthread
pg@chairperson-HP:~$ ./a.out

 The current Thread Scope=>0
 The Current PTHREAD_SCOPE_PROCESS=>1
 The Current PTHREAD_SCOPE_SYSTEM=>0
 Currently the Scope is=> System Contention Scope
 The current Thread Scope after Change=>0
```

OPERATING SYSTEMS

Thread Scheduling - pthread example

```c
// C program Maximum Threads in a processs
// On Success, pthread_create returns 0 and
// on Error, it returns the error number
// Program has to be compilied using
// gcc Threads_Scheduling.c -pthread
// Because pthread has to be externally linked else results linker error

// Linux supports PTHREAD_SCOPE_SYSTEM, but not PTHREAD_SCOPE_PROCESS.


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

#define Threads_Max 5

int j;

int ThreadScope;

int Count = 0;

pthread_t Threads[Threads_Max];

pthread_attr_t ThreadAttributes;


void *ThreadFunction ( void *TParameter)
{

        Count++;
        pthread_exit(0);
}
```

OPERATING SYSTEMS

Thread Scheduling - pthread example

```c
int main(int argc, char **argv)
{

    int i;


    pthread_attr_init(&ThreadAttributes);

    ThreadScope = pthread_attr_getscope(&ThreadAttributes,&ThreadScope);

    if(ThreadScope!=0)
        fprintf(stderr,"\nUnable to get scheduling scope information");

    if(ThreadScope==PTHREAD_SCOPE_PROCESS)
        fprintf(stdout,"\n Currently the Scope is=> Process Contention Scope");
    else
        fprintf(stdout,"\n Currently the Scope is=> System Contention Scope");


    printf("\n\n\n\n");
    ThreadScope = pthread_attr_setscope(&ThreadAttributes,PTHREAD_SCOPE_SYSTEM);

    /* create the threads */
    for (i = 0; i < Threads_Max; i++)
    {

        pthread_create(&Threads[i],&ThreadAttributes,ThreadFunction,NULL);
        printf("\n Currently Thread =>%p is running with Count at =>%d", Threads[i],Count) ;

    }

    /* now join on each thread */
    for (i = 0; i < Threads_Max;i++)
        pthread_join(Threads[i],NULL);

    printf("\n The state of Count After Joining is =>%d",Count);
    printf("\n\n\n\n");
}
```

```
pg@chairperson-HP:~$ ./a.out

Currently the Scope is=> System Contention Scope




Currently Thread =>0x7f4e90665700 is running with Count at =>0
Currently Thread =>0x7f4e8fe64700 is running with Count at =>1
Currently Thread =>0x7f4e8f663700 is running with Count at =>2
Currently Thread =>0x7f4e8ec4a700 is running with Count at =>2
Currently Thread =>0x7f4e8e449700 is running with Count at =>4
The state of Count After Joining is =>5
```

FCFS Scheduling

```c
#include<stdio.h>

int main()

{
    int n,bt[20],wt[20],tat[20];
    int i,j;
    float avwt=0,avtat=0;
    printf("\nEnter total number of processes(maximum 20):");
    scanf("%d",&n);

    printf("\nEnter Process Burst Timen");
    for(i=0;i<n;i++)
    {
        printf("P[%d]:",i+1);
        scanf("%d",&bt[i]);
    }

    wt[0]=0;

    for(i=1;i<n;i++)
    {
        wt[i]=0;
        for(j=0;j<i;j++)
            wt[i]+=bt[j];
    }

    printf("\n\tProcesst\tBurst Time\tWaiting Time\tTurnaround Time");

    for(i=0;i<n;i++)
    {
        tat[i]=bt[i]+wt[i];
        avwt+=wt[i];
        avtat+=tat[i];
        printf("\n\tP[%d]\t\t\t%d\t\t%d\t\t%d",i+1,bt[i],wt[i],tat[i]);
    }

    avwt/=n;
    avtat/=n;
    printf("\nAverage Waiting Time:%f",avwt);
    printf("\nAverage Turnaround Time:%f",avtat);
    printf("\n\n\n");

    return 0;
}
```

```
pg@chairperson-HP:~$ ./a.out

Enter total number of processes(maximum 20):5

Enter Process Burst TimenP[1]:1
P[2]:2
P[3]:3
P[4]:4
P[5]:5

        Processt        Burst Time      Waiting Time    Turnaround Time
        P[1]                    1               0               1
        P[2]                    2               1               3
        P[3]                    3               3               6
        P[4]                    4               6               10
        P[5]                    5               10              15
Average Waiting Time:4.000000
Average Turnaround Time:7.000000
```

THANK YOU

Nitin V Pujari
Faculty, Computer Science
Dean - IQAC, PES University

nitin.pujari@pes.edu

For Course Deliverables by the Anchor Faculty click on www.pesuacademy.com