Lab 9
Thread Management

TA:Cheng-Chia Chang

Professor: Hsung-Pin Chang

Operating System Lab

Outline

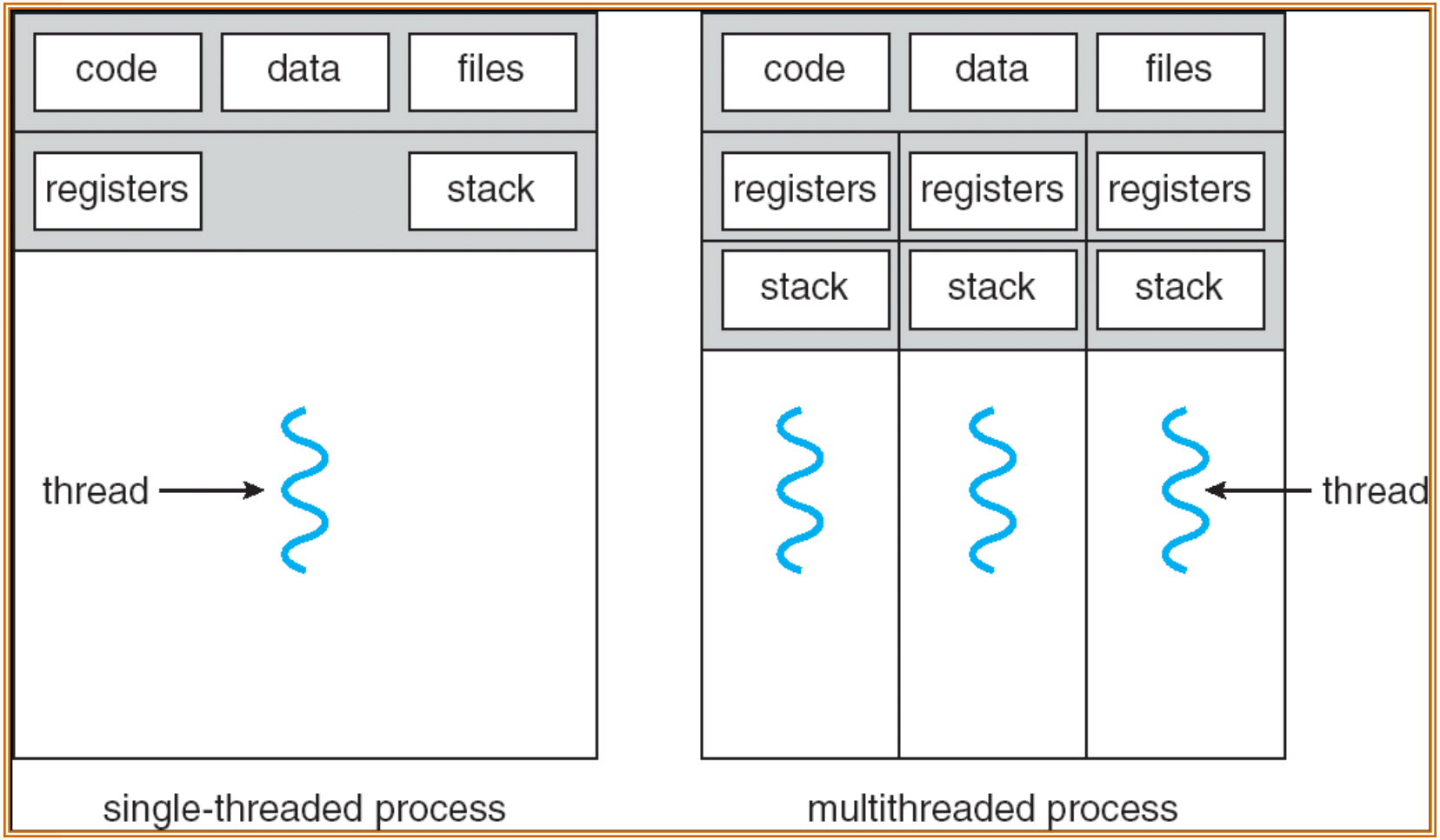
- **Thread Overview**
- **Pthreads**
  - Creating threads
  - Exit threads
  - Join threads
- **Java Thread**
  - Install the Java Virtual Machine
  - Creating Threads
  - Excuting Threads
  - Join threads
- **Exercise**

Thread Overview

Why We Need Threads?

- **A traditional process**
  - Needs a fair amount of "overhead“
  - Has a single thread of control
- **If the process has multiple threads of control, it can do more than one task at a time.**
- **Threads**
  - Run as independent entities largely because they duplicate only the bare essential resources that enable them to exist as executable code

What Is a Thread?



What Is a Thread? (cont.)

- A thread is a flow of control within a process.
- A multithreaded process contains several different flows of control within the same address space.
- A thread has a thread ID, a program counter, a register set, and a stack, and is similar to a process has.
- However, a thread *shares* with other threads in the same process its *code section*, *data section*, and other OS resources.
  - Thus, thread is *lightweight*

The Benefits of Multithreaded Programming

- **Responsiveness**
  - A process can still execute even if some part of process is paused or taking a long time.
- **Resource sharing**
  - Share resources of the process to which they belong.
- **Economy**
  - Allocating memory and resources for process creation is costly.
  - Threads that share resources of the process to which they belong is more economical to *create* and *context switch* threads.
- **Utilization of multiprocessor architectures**
  - Be greatly increased in a multiprocessor architecture.

Challenges When Using Thread

- **Because threads within the same process *share resources*:**
  - Changes made by one thread to shared system resources will be seen by all other threads.
  - Reading and writing to *the same memory locations* at *the same time* by two different thread is possible
    - *Race condition*
    - Therefore, thread needs explicit ***synchronization*** by the programmer.

POSIX Thread

What Are Pthreads?

- Historically, hardware vendors have implemented their own proprietary versions of threads .
  - Difficult for programmers to develop portable threaded applications.
- For UNIX systems, a standardized programming interface has been specified by the IEEE POSIX 1003.1c standard (1995).
- Implementations which adhere to this standard are referred to as *POSIX threads*, or *Pthreads*.
- Pthreads are defined as a set of C language programming types and procedure calls, implemented with a *pthread.h* header/include file and a thread library.

The Pthreads API

- There is a whole set of library calls associated with threads, most of whose names start with `pthread_`.
- To use these library calls, we must include the file `pthread.h`, and link with the pthread library using `-pthread`.

POSIX function	description
pthread_cancel	terminate another thread
pthread_create	create a thread
pthread_detach	set thread to release resources
pthread_equal	test two thread IDs for equality
pthread_exit	exit a thread without exiting process
pthread_kill	send a signal to a thread
pthread_join	wait for a thread
pthread_self	find out own thread ID

Referencing Threads by ID

- **`pthread_self`**: A thread can find out its ID.

```
#include<pthread.h>
pthread_t pthread_self(void);
```

- **`pthread_equal`**: To compare thread IDs for equality.

```
#include<pthread.h>
int pthread_equal(pthread_t t1,pthread_t t2);

/*Return a nonzero value if t1 equals t2,
  0 if the thread IDs are not equal*/
```

Creating a Thread

- **`pthread_create`**: creates a thread.

```
#include<pthread.h>
int pthread_create(pthread_t *thread,
        const pthread_att_t *attr,
        void *(*start_routine)(void *), void *arg);
/*Return 0 if successful*/
```

thread: *Points to the ID of the newly created thread.*

attr: *An attribute object that encapsulates the attributes of a thread. NULL for the default values.*

start_routine: *The C routine that the thread calls when it begins execution*

arg: *The argument that is to be passed to start_routine. NULL may be used if no argument is to be passed.*

Exiting

- If the main thread has no work to do after creating other threads, it should either block until all threads have completed or call `pthread_exit()`.

```
#include<pthread.h>
pthread_t pthread_exit(void *value_ptr);
```

Joining

- **`pthread_join`**: causes the caller to wait for the specified thread to exit.

```
#include<pthread.h>
pthread_t pthread_join(pthread_t thread,
                       void **value_ptr);
/*Return 0 if successful*/
```

thread: The *ID of the terminating thread*.

value_ptr: *Provides a location for a pointer to the return status that the target thread passes to pthread_exit. The caller does not retrieve the target thread return status if NULL.*

Example

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

void *PrintHello(void *arg)
{
    printf("This is Hello THread\n");
    printf("Thread ID:%lu\n",pthread_self());
    printf("Argument:%d\n",(int)arg);
    int re = 50;
    pthread_exit(&re);
}

int main()
{
    pthread_t thread;
    int rc, t = 100;
    void *ret;
    rc = pthread_create(&thread, NULL, PrintHello, &t);
    if (rc){
        printf("CREATE ERROR");
        exit(-1);
    }

    rc = pthread_join(thread, &ret);
    if (rc){
        printf("JOIN ERROR");
        exit(-1);
    }
    printf("return value:%d\n",(int) ret);
    return 0;
}
```

Compile

```
檔案(F)  編輯(E)  檢視(V)  終端機(T)  求助(H)
scribe@osnet-scribe-ubuntu64:~/Documents/oslab/chap8$ gcc -o example example.c
example.c: In function ‘PrintHello’:
example.c:8: warning: cast from pointer to integer of different size
example.c: In function ‘main’:
example.c:15: warning: cast to pointer from integer of different size
/tmp/cc1Egf0Y.o: In function `main':
example.c:(.text+0x83): undefined reference to `pthread_create'
example.c:(.text+0x94): undefined reference to `pthread_join'
collect2: ld returned 1 exit status
scribe@osnet-scribe-ubuntu64:~/Documents/oslab/chap8$ ls
example.c
scribe@osnet-scribe-ubuntu64:~/Documents/oslab/chap8$ gcc -o example example.c -
pthread
example.c: In function ‘PrintHello’:
example.c:8: warning: cast from pointer to integer of different size
example.c: In function ‘main’:
example.c:15: warning: cast to pointer from integer of different size
scribe@osnet-scribe-ubuntu64:~/Documents/oslab/chap8$ ls
example  example.c
scribe@osnet-scribe-ubuntu64:~/Documents/oslab/chap8$
```

- While compiling pthread program, ensure to add “-pthread” option to command line. (Link libpthread.a library)

Example (cont.)

Applications Places System oslab

oslab@oslab-desktop: ~/OSLAB/Lab8_Thread_Management/example

File Edit View Terminal Help

```
oslab@oslab-desktop:~/OSLAB/Lab8_Thread_Management/example$ ./thread_example
This is Hello Thread
Thread ID:3085982608
Argument: 100
oslab@oslab-desktop:~/OSLAB/Lab8_Thread_Management/example$
```

oslab@oslab-desktop: ...

Java Thread

Install the Java Virtual Machine

```
test@testing:~$ sudo add-apt-repository ppa:openjdk-r/ppa
[sudo] password for test:

 More info: https://launchpad.net/~openjdk-r/+archive/ubuntu/ppa
Press [ENTER] to continue or ctrl-c to cancel adding it

gpg: keyring `/tmp/tmp7xdicl02/secring.gpg' created
gpg: keyring `/tmp/tmp7xdicl02/pubring.gpg' created
gpg: requesting key 86F44E2A from hkp server keyserver.ubuntu.com
gpg: /tmp/tmp7xdicl02/trustdb.gpg: trustdb created
gpg: key 86F44E2A: public key "Launchpad OpenJDK builds (all archs)" imported
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)
OK
```

Install the Java Virtual Machine (Cont.)

```
test@testing:~$ sudo apt-get update
Hit:1 http://tw.archive.ubuntu.com/ubuntu xenial InRelease
Hit:2 http://tw.archive.ubuntu.com/ubuntu xenial-updates InRelease
Hit:3 http://tw.archive.ubuntu.com/ubuntu xenial-backports InRelease
Get:4 http://ppa.launchpad.net/openjdk-r/ppa/ubuntu xenial InRelease [17.5 kB]
Hit:5 http://security.ubuntu.com/ubuntu xenial-security InRelease
Get:6 http://ppa.launchpad.net/openjdk-r/ppa/ubuntu xenial/main i386 Packages [1
1.6 kB]
Get:7 http://ppa.launchpad.net/openjdk-r/ppa/ubuntu xenial/main Translation-en [
1904 B]
Fetched 31.0 kB in 3s (8602 B/s)
Reading package lists... Done
```

Install the Java Virtual Machine (Cont.)

```
test@testing:~$ sudo apt install openjdk-11-jdk
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  ca-certificates-java fonts-dejavu-extra java-common libatk-wrapper-java
  libatk-wrapper-java-jni libgif7 libice-dev libpthread-stubs0-dev
  libreoffice-avmedia-backend-gstreamer libreoffice-base-core libreoffice-calc
  libreoffice-core libreoffice-draw libreoffice-gnome libreoffice-gtk
  libreoffice-impress libreoffice-math libreoffice-ogltrans
  libreoffice-pdfimport libreoffice-writer libsm-dev libx11-dev libx11-doc
  libxau-dev libxcb1-dev libxdmcp-dev libxt-dev openjdk-11-jdk-headless
  openjdk-11-jre openjdk-11-jre-headless python3-uno x11proto-core-dev
  x11proto-input-dev x11proto-kb-dev xorg-sgml-doctools xtrans-dev
Suggested packages:
  default-jre libice-doc libreoffice-base ocl-icd-libopencl1
  libreoffice-evolution fonts-crosextra-caladea fonts-crosextra-carlito
  libreoffice-gcj libreoffice-java-common libsm-doc libxcb-doc libxt-doc
  openjdk-11-demo openjdk-11-source visualvm fonts-ipafont-gothic
  fonts-ipafont-mincho fonts-wqy-microhei | fonts-wqy-zenhei fonts-indic
The following NEW packages will be installed:
  ca-certificates-java fonts-dejavu-extra java-common libatk-wrapper-java
  libatk-wrapper-java-jni libgif7 libice-dev libpthread-stubs0-dev libsm-dev
  libx11-dev libx11-doc libxau-dev libxcb1-dev libxdmcp-dev libxt-dev
  openjdk-11-jdk openjdk-11-jdk-headless openjdk-11-jre
  openjdk-11-jre-headless x11proto-core-dev x11proto-input-dev x11proto-kb-dev
  xorg-sgml-doctools xtrans-dev
The following packages will be upgraded:
  libreoffice-avmedia-backend-gstreamer libreoffice-base-core libreoffice-calc
  libreoffice-core libreoffice-draw libreoffice-gnome libreoffice-gtk
  libreoffice-impress libreoffice-math libreoffice-ogltrans
  libreoffice-pdfimport libreoffice-writer python3-uno
13 upgraded, 24 newly installed, 0 to remove and 243 not upgraded.
Need to get 321 MB of archives.
After this operation, 435 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

Install the Java Virtual Machine (Cont.)

```
test@testing:~$ java -version
openjdk version "11.0.4" 2019-07-16
OpenJDK Runtime Environment (build 11.0.4+11-post-Ubuntu-116.04.1)
OpenJDK Server VM (build 11.0.4+11-post-Ubuntu-116.04.1, mixed mode, sharing)
```

Creating Thread

- **There are two techniques for explicitly creating threads in a Java program.**
  - **To derive from the Thread class and to override its run() method.**
  - **To define a class that implements the Runnable interface.**

Creating Thread(cont.)

- Method 1: **override run() method.**

```
public class Hello1 extends Thread
{
        String name;

        public Hello1(String n)
        {
                name = n;

        }

        public void run()
        {
                for(int i = 1; i <= 10; i++)
                        System.out.println(name+" Hello "+i);

        }
}
```

Creating Thread(cont.)

- Method 2: **Implements the Runnable interface**

```
public class Hello2 implements Runnable
{
        String name;

        public Hello2(String n)
        {
                name = n;
        }

        public void run()
        {
                for(int i = 1; i <= 10; i++)
                        System.out.println(name+" Hello "+i);
        }
}
```

Excuting Thread

- (Thread name)**.start**

```
public static void main(String argv[])
{

        Hello2 h1 = new Hello2("thread1");
        Thread t1 = new Thread(h1);

        t1.start();
}
```

Joining

- (Thread name).**join**

```java
public class add implements Runnable
{
    int sum = 0;
    public void run()
    {
        for(int i = 1; i <= 10; i++)
            sum += 1;

        printThread h1= new printThread(sum);
        Thread t_print = new Thread(h1);
        t_print.start();

        try{
            t_print.join();
        }
        catch(InterruptedException ie)
        {
            System.err.println(ie);
        }

        System.out.println("exit add thread");

    }
}
```

add.java [+]

```java
public class join
{
    public static void main(String argv[])
    {
        Thread t_add = new Thread(new add());
        t_add.start();

    }

}
```

```java
public class printThread implements Runnable
{
    int sum = 0;

    public printThread(int num)
    {
        sum = num;
    }

    public void run()
    {
        for(int i = 1; i <= 10; i++)
            sum += 1;
        System.out.println("First print sum:"+sum);
    }

}
```

printThread.java

Compiler & Excute

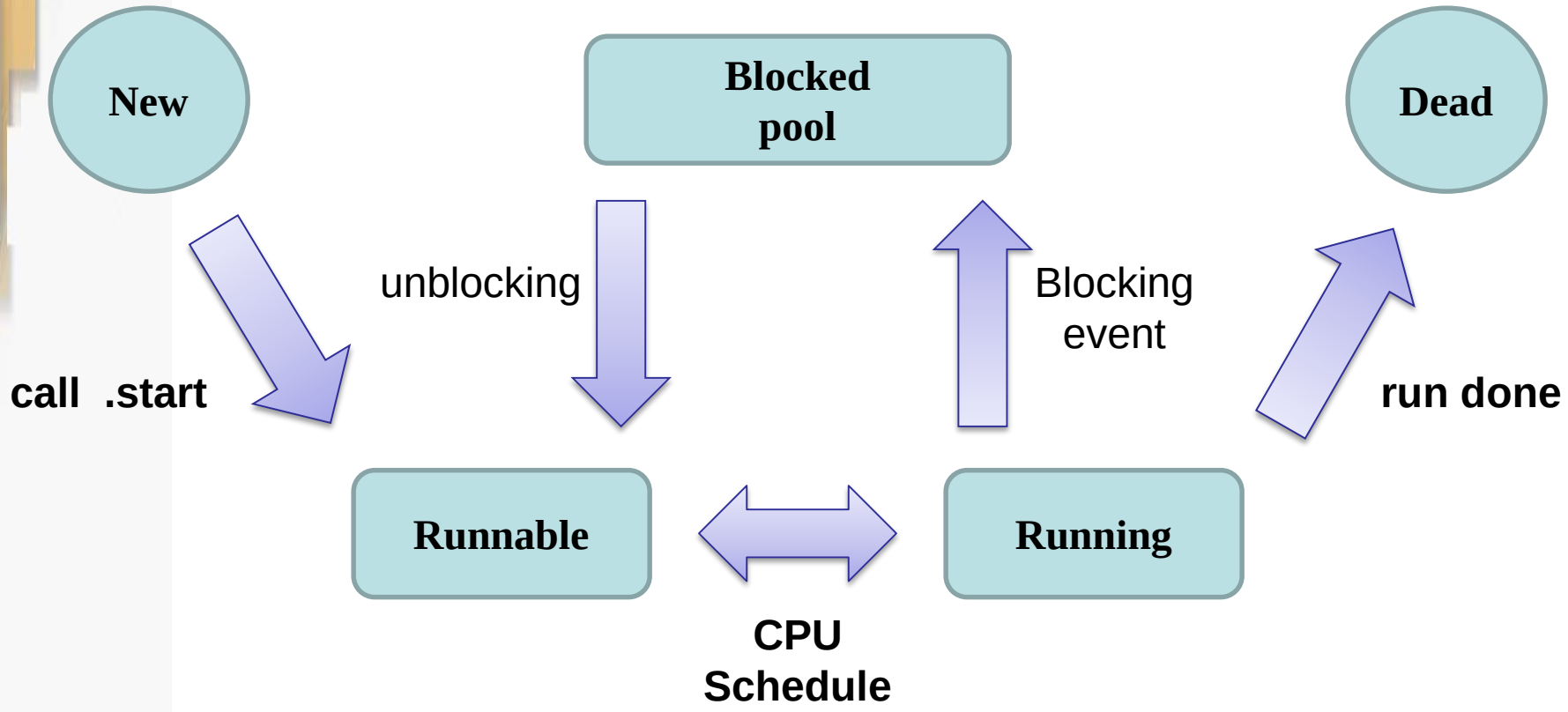
- Compiler

```
arthur@arthur-VirtualBox:~/oslab/lab09/lab09_example$ javac ThreadEx.java
arthur@arthur-VirtualBox:~/oslab/lab09/lab09_example$
```

- Excute java

```
arthur@arthur-VirtualBox:~/oslab/lab09/lab09_example$
arthur@arthur-VirtualBox:~/oslab/lab09/lab09_example$ java ThreadEx
```

Thread state diagram



Example

```java
public class ThreadEx
{
    public static void main(String argv[])
    {
        ShareData s1 = new ShareData();
        ShareData s2 = new ShareData();

        Thread t1 = new Thread(s1);
        Thread t2 = new Thread(s2);

        t1.start();
        t2.start();

    }

}

```

```java
public class ShareData implements Runnable
{
    int i = 0;

    public void run()
    {
        while(i < 10){
            i++;
            System.out.println(Thread.currentThread().getName()+":"+i);
        }
    }

}
```

Example(cont.)

```
arthur@arthur-VirtualBox:~/oslab/lab09/lab09_example$ java ThreadEx
Thread-1:1
Thread-1:2
Thread-1:3
Thread-1:4
Thread-1:5
Thread-1:6
Thread-1:7
Thread-1:8
Thread-1:9
Thread-1:10
Thread-0:1
Thread-0:2
Thread-0:3
Thread-0:4
Thread-0:5
Thread-0:6
Thread-0:7
Thread-0:8
Thread-0:9
Thread-0:10
```

Example(cont.)

```java
public class ThreadEx
{
    public static void main(String argv[])
    {
        ShareData s = new ShareData();

        Thread t1 = new Thread(s);
        Thread t2 = new Thread(s);

        t1.start();
        t2.start();

    }

}
```

```
arthur@arthur-VirtualBox:~/oslab/lab09/lab09_example$ java ThreadEx
Thread-1:2
Thread-1:3
Thread-1:4
Thread-1:5
Thread-1:6
Thread-1:7
Thread-1:8
Thread-1:9
Thread-1:10
Thread-0:1
```

Exercise

Lab1(20pts.)

- **Using Pthreads to create four identical threads**
  - Also declare a global variable **`count`** = 0.
  - The main program passes a value containing the number of iterations to the threads.
- **Each thread increments the same global variable 250,000 times**
  - Each thread prints the number of iterations from argument and its thread ID.
  - So `count` is incremented a total of 1,000,000 times.

Lab1 (cont.)

- **The main program waits for the four threads to complete**
  - Then print out the global **`count`** variable at last.
- **Example:**

```
$ ./lab1
Thread 1: ID 4073392 Completed.
Thread 0: ID 4073072 Completed.
Thread 3: ID 4078520 Completed.
Thread 2: ID 4078200 Completed.
Value = 1000000
```

Lab 1-1(20pts.)

- **Run your Lab1 program several times**
- **Questions:**
  - Observe all of the results you got, and think about what problem does it have? (10pts.)
  - Compare to fork() process by doing all the same thing to the global variable, can you figure out what's the difference between them? (10pts.)

Lab2(40pts.)

- **Using Java thread to create four identical threads**
  - Also declare a global variable **`count`** = 0.
  - The main program passes a value containing the number of iterations to the threads.
- **Each thread increments the same global variable 250,000 times**
  - Each thread prints the number of iterations from argument and its thread ID.
  - So `count` is incremented a total of 1,000,000 times.

Lab2 (cont.)

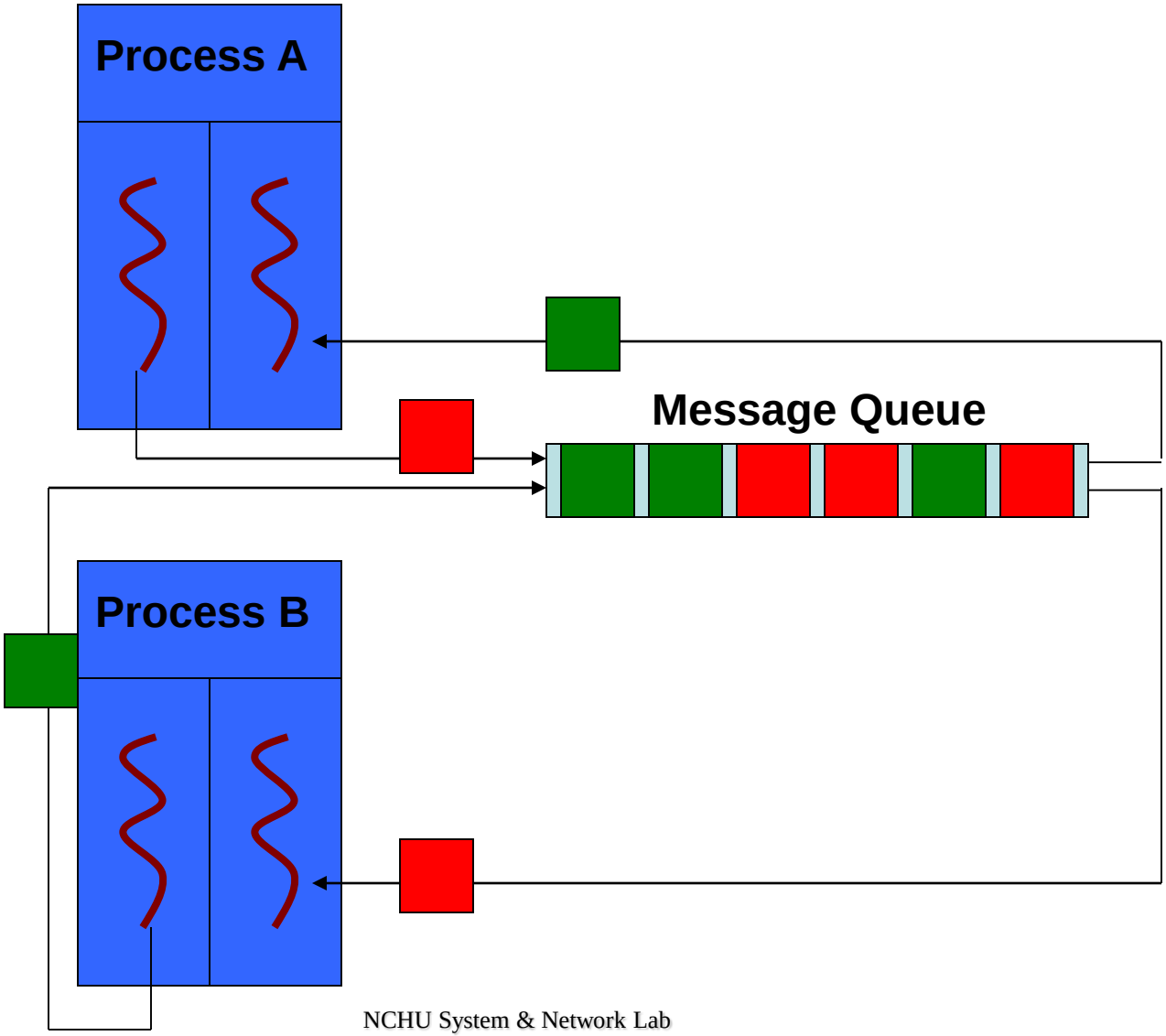
- **The main program waits for the four threads to complete**
  - Then print out the global **`count`** variable at last.
- **Example:**

```
$ ./lab1
Thread 1: ID 4073392 Completed.
Thread 0: ID 4073072 Completed.
Thread 3: ID 4078520 Completed.
Thread 2: ID 4078200 Completed.
Value = 1000000
```

Lab 3 (10pts.)

- **Using the message queue which we learned last week**
- **Create two processes, each of them has two threads**
  - One thread can allow user type messages iteratively and send those messages to a queue.
  - The other one can receive the messages which be sent to the queue by the other process and print on screen.
- **The two processes use the same message queue, but assign different message type to distinguish each other**

Lab3 (cont.)



Lab3 Result Example

```
$ ./lab2_A
Lab2_A_Process

Enter some text: Hello This is A.
Enter some text:
Received: Hello, I'm B.


Received: Nice to meet you

me too
Enter some text: bye
Enter some text:
Received: 881


Received: exit

exit
```

```
$ ./lab2_B
Lab2_B_Process

Enter some text:
Received: Hello This is A.

Hello, I'm B.
Enter some text: Nice to meet you
Enter some text:
Received: me too


Received: bye

881
Enter some text: exit

Received: exit
```

References

- Avi Silberschatz, Peter Baer Galvin and Greg Gagne, "**Operating System Concepts**," John Wiley & Sons, 6th Edition, 2001
- **POSIX thread programming**

  http://www.llnl.gov/computing/tutorials/pthreads/
- "**Unix Systems Programming: Communication, Concurrency, and Threads**" by Kay A. Robbins, Steven Robbins
- Neil Matthew and Richard Stones, "**Beginning Linux Programming**," Wiley publishing, 3rd Edition, 2004
- W. Richard Stevens, "**Advanced Programming in the UNIX Environment**," Addison-Wesley, 1992

References (cont.)

- Marshall Brain, "**Win32 System Services: The Heart of Windows 95 and Windows NT**," Prentice Hall PTR, 2nd Edition, 1996
- http://publib.boulder.ibm.com/infocenter/iseries/v5r3/index.jsp?topic=/rzahw/rzahwsemco.htm
- https://stackoverflow.com/questions/52504825/how-to-install-jdk-11-under-ubuntu