Bach Khoa University Of Technology
Faculty of Computer Science & Engineering

Course: Parallel Processing
Lab #1&2 - Introduction & Multithreads

Minh Thanh Chung

January 22, 2018

Goal: This lab helps student to get familiar with experimental environment on the server - SuperNode-XP. Then, student can review the notation of Thread in Operating System Course and program the multithread program.

Content: Section 1 introduces how to access the server to do examples and exercises. The next Sections show example to practice with multithread program using POSIX Threads and OpenMP.

Result: After this Lab, student can understand and write a program with multithread in parallel.

Contents

1 Introduction

1.1 Contents and Grading

1.2 How to access the server of this course

In this course, each student has an account to access the server and do examples or exercises on it. However, if you want to do the Lab on your laptop, you can set up your own environment with Virtual Machine. Because all of Labs and Assignment will be tested on Linux OS.

For Windows: a simple tool that you can use is Putty (`https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html`). Information of gateway to access from outside networks:

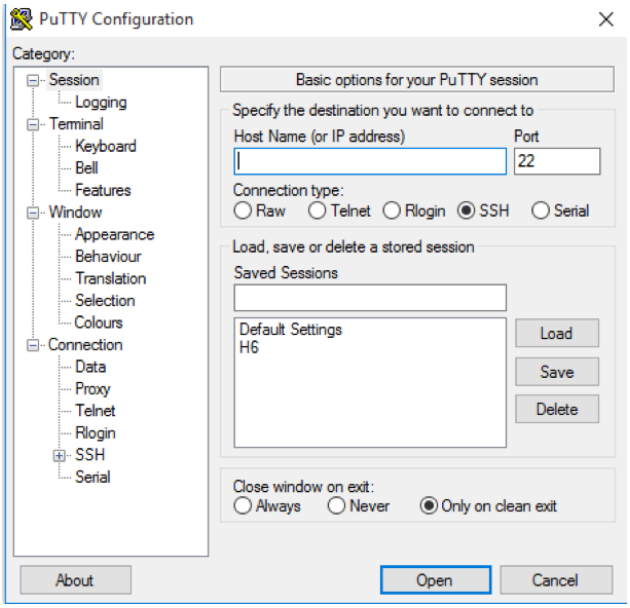

Figure 1: GUI of Putty

- Hostname: **hpcc.hcmut.edu.vn**
- Port: 22

After that, click open. Then, you need an account and password to access the gateway (this is a public account):

- Username: **std**
- Password: **bkhpcc**

At this step, you are just in the gateway, then, you need to access your machine by ***ssh*** protocol.

```
$ ssh username@IP_address
```

For Linux, you do not need to use other tools, you can access by ***ssh*** protocol.

```
$ ssh std@hpcc.hcmut.edu.vn

// Then, you continuously ssh to the machine
$ ssh username@IP_address
```

1.3 How to transfer data between host and server

For Windows, you also use a tool to copy data between server and host, WinSCP (`https://winscp.net/eng/download.php`). Information for data transfer (with the window on the left hand side):

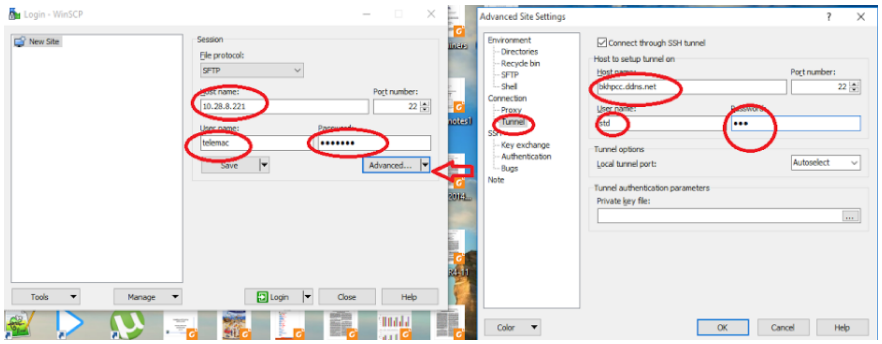

Figure 2: Data transfer between host and server by WinSCP

- Hostname: **IP_address** (of your machine)
- Username: **user_name**
- Password: **pass**

However, to transfer the data, you need pass over the gateway, therefore, in the tab Advanced you need to fill some info such as (note, choose Tunnel at Connection Tab):

- Hostname: **hpcc.hcmut.edu.vn** (address of gateway)
- Username: **std**
- Password: **bkhpcc**

For Linux, you also can copy data by command line:

```
$ ssh -L 1234:10.1.6.1:22 std@hpcc.hcmut.edu.vn cat -
$ scp -P 1234 1510180@127.0.0.1:/home/1510180/a.txt /home/
```

1.4 Set up Virtual Machine on your laptop [Optional]

2 Review

2.1 POSIX Threads - Linux

What is Pthreads?
Historically, hardware vendors have implemented their own proprietary versions of threads. These implementations differed substantially from each other making it difficult for programmers to develop portable threaded applications. The POSIX standard has continued to evolve and undergo revisions, including the Pthreads specification.

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 - though this library may be part of another library, such as libc, in some implementations.

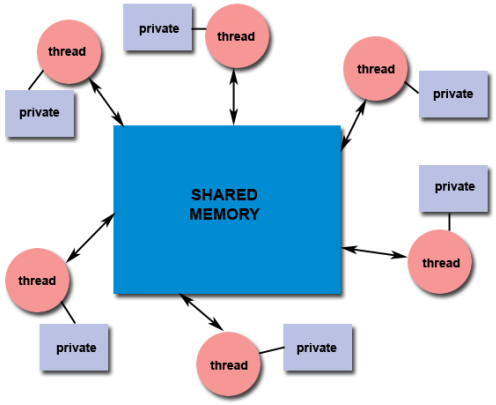

Figure 3: Shared Memory Model

All threads have access to the same global, shared memory. Threads also have their own private data. Programmers are responsible for synchronizing access (protecting) globally shared data.

2.2 Examples

Compiling Threaded Programs: several examples of compile commands used for pthreads codes are listed in the table below.

Table 1: Command lines for compiling Threaded programs

Compiler / Platform	Compiler Command	Description
INTEL Linux	icc -pthread	C
	icpc -pthread	C++
PGI Linux	pgcc -lpthread	C
	pgCC -lpthread	C++
GNULinux, Blue Gene	gcc -pthread	GNU C
	g++ -pthread	GNU C++

Example1: Pthread Creation and Termination
This simple example code creates 5 threads with the pthread_create() routine. Each thread prints a "Hello World!" message, and then terminates with a call to pthread_exit().

```
#include <pthread.h>
#include <stdio.h>
#define NUM_THREADS 10

// user-defined functions
void * user_def_func(void *threadID){
    long TID;
    TID = (long) threadID;
    printf("Hello World! from thread #%ld\n", TID);
    pthread_exit(NULL);
}

int main(int argc, char *argv){
    pthread_t threads[NUM_THREADS];
    int create_flag;
```

```
    long i;
    for(i = 0; i < NUM_THREADS; i++){
        printf("In main: creating thread %ld\n", i);
        create_flag = pthread_create(&threads[i], NULL, user_def_func, (void *)i);
        if(create_flag){
            printf("ERROR: return code from pthread_create() is %d\n", create_flag);
            exit(-1);
        }
    }

    // free thread
    pthread_exit(NULL);
    return 0;
}
```

Example2: Thread Argument Passing

This code fragment demonstrates how to pass a simple integer to each thread. The calling thread uses a unique data structure for each thread, insuring that each thread's argument remains intact throughout the program.

```
...
/* Thread Argument Passing */
// case-study 1
long taskids[NUM_THREADS];

// case-study 2
...

int main (int argc, char *argv[]){
    pthread_t threads[NUM_THREADS];
    int creation_flag;
    long i;
    for(i = 0; i < NUM_THREADS; i++){
        // pass arguments
        taskids[i] = i;
        printf("In main: creating thread %ld\n", i);
        creation_flag = pthread_create(&threads[i], NULL, user_def_func, (void *)taskids[i]);
          ...
    }

    ...
}
```

Question: how to setup/pass multiple arguments via a structure?

3 Multithread Programming with OpenMP

3.1 Motivation

What is OpenMP?

- An Application Program Interface (API) that may be used to explicitly direct **multithreaded**, **shared memory** parallelism.
- Comprised of three primary API components:
  - Compiler Directives
  - Runtime Library Routines
  - Environment Variables

Goals of OpenMP

- Standardization
- Lean and Mean
- Ease of Use
- Portability

Shared Memory Model OpenMP is designed for multi-processor/core, shared memory machines. The underlying architecture can be shared memory UMA or NUMA.

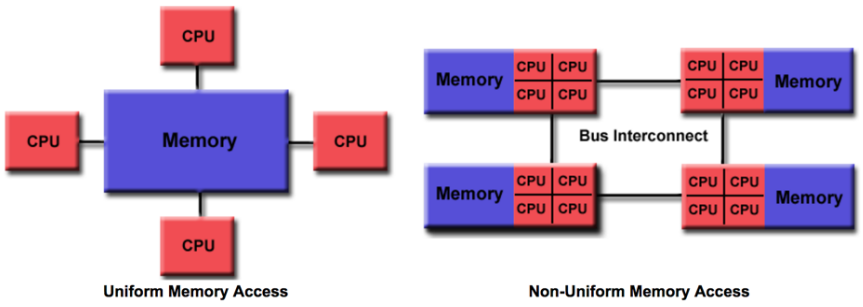

Figure 4: Shared Memory Model for OpenMP

3.2 Examples

Example1: Simple "Hello World" program. Every thread executes all code enclosed in the parallel region. OpenMP library routines are used to obtain thread identifiers and total number of threads.

```
#include <omp.h>

int main(int argc, char *argv[]) {

  int nthreads, tid;

 /* Fork a team of threads with each thread having a private tid variable */
 #pragma omp parallel private(tid)
  {
```

```
10
    /* Obtain and print thread id */
    tid = omp_get_thread_num();
    printf("Hello World from thread = %d\n", tid);

15  /* Only master thread does this */
    if (tid == 0)
     {
     nthreads = omp_get_num_threads();
     printf("Number of threads = %d\n", nthreads);
20   }

    }  /* All threads join master thread and terminate */
  }
```

4 Exercises

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book.

5 Submission

5.1 Source code

Requirement: you have to code the system call followed by the coding style. Reference: https://www.gnu.org/prep/standards/html_node/Writing-C.html.
After you finish the assignment, moving your report to source code directory and compress the whole directory into a single file name `assignment1_MSSV.zip` and submit to Sakai.