OBL2-OS

October 25, 2018

This is a mandatory assignment. Use resources from the course to answer the following questions. Take care to follow the numbering structure of the assignment in your submission. Some questions may require a little bit of web searching. Some questions require you to have access to a Linux machine, for example running natively or virtually on your own PC, or by connecting to gremlin.stud.iie.ntnu.no over SSH (Secure Shell). Working in groups is permitted, but submissions must be individual.

1 Processes and threads

1. Explain the difference between a process and a thread.

A process is a series of instructions. A thread is an individual path of execution within a process.

2. Describe a scenario where it is desirable to:

• Write a program that uses threads for parallel processing

In a GUI.

• Write a program that uses processes for parallel processing

In an OS.

1. Explain why each thread requires a thread control block (TCB).

Because each thread needs to store specific information like a PC, tid, thread’s stack in the process, thread state, register values, pointer to the pcb of the thread’s process.

1. What is the difference between cooperative (voluntary) threading and pre-emptive (involuntary) thread- ing? Briefly describe the steps necessary for a context switch for each case.

Cooporative: once a thread is given control, it continues to run until it is finished. Pre-emptive: the machine can take control from the thread at any time.

2 C program with POSIX threads

\*nix operating systems use POSIX threads, which are provided by the pthread library. Consider the following adapted code from the textbook (the code has been modified slightly to use pthread, while the book assumes its own thread implementation).

#include <stdio.h>

#include <pthread.h>

#define NTHREADS 10

pthread\_t threads[NTHREADS];

void \*go (void \*n) {

printf("Hello from thread %ld\n", (long)n);

pthread\_exit(100 + n);

// REACHED?

}

int main() {

long i;

for (i = 0; i < NTHREADS; i++) pthread\_create(&threads[i], NULL, go, (void\*)i);

for (i = 0; i < NTHREADS; i++) {

long exitValue;

pthread\_join(threads[i], (void\*)&exitValue);

printf("Thread %ld returned with %ld\n", i, exitValue);

}

printf("Main thread done.\n");

return 0;

}

We can compile the code and tell the compiler to link the pthread library:

$ gcc -o threadHello threadHello.c -lpthread

At the command prompt, run the program using ./threadHello. The program gives output similar to the following:

Hello from thread 0

Hello from thread 3

Hello from thread 5

Hello from thread 1

Hello from thread 4

Thread 0 returned with 100

Thread 1 returned with 101

Hello from thread 9

Hello from thread 8

Hello from thread 2

Hello from thread 7

Hello from thread 6

Thread 2 returned with 102

Thread 3 returned with 103

Thread 4 returned with 104

Thread 5 returned with 105

Thread 6 returned with 106

Thread 7 returned with 107

Thread 8 returned with 108

Thread 9 returned with 109

Main thread done.

Study the code and the output. Run the code several times. Answer the following questions.

1. Which part of the code (e.g., the task) is executed when a thread runs? Identify the function and describe briefly what it does.

The function is called go. It prints out the current thread and exits.

2. Why does the order of the “Hello from thread X” messages change each time you run the program?

Because the system environment changes causing each thread to use more or less time to finish.

1. What is the minimum and maximum number of threads that could exist when thread 8 prints “Hello”?

Minimum 2 (main + thread 8), maximum 11 (main + all threads).

1. Explain the use of pthread join function call.

It waits for termination of the calling thread.

5. What would happen if the function go is changed to behave like this:

void \*go (void \*n) {

printf("Hello from thread %ld\n", (long)n);

if(n == 5)

sleep(2); // Pause thread 5 execution for 2 seconds

pthread\_exit(100 + n);

// REACHED?

}

Thread 5 would live 2 seconds longer.

6. When pthread join returns for thread X, in what state is thread X?

It will be terminated.

Page 2