COMP3511 Project #1: Nachos and Thread

In this project, you will learn how to create threads in Nachos. You are given a simple thread system for Nachos. Here are your tasks:

1. Compile Nachos and run the system.
2. Add some lines to the Nachos code. Recompile and run it.
3. Add several specified lines in the Nachos code. Recompile and run it.
4. Save the output and explain the results.

Don't be overwhelmed by the sheer amount of code provided. In fact you don't need to care about most of it. The part that you need to read or modify will be given in this instruction.

Task 1: Running Nachos with pre-implemented semaphore

Step 1: download Nachos source code, with pre-implemented semaphore.

```
wget http://course.cse.ust.hk/comp3511/project/project1/os_nachos_proj1.tar.gz
```

Step 2: Extract the source code.

```
tar -zxvf os_nachos_proj1.tar.gz
```

Step 3: Compile the code

Enter the folder "os_nachos_ proj1" and then run "make"

Step 4: Run nachos

Run "./nachos"

In this program, Nachos creates one thread, which does nothing except telling us its name "Thread1" and its termination.

If you succeed in running nachos, you will see these messages:

```
Hello, my name is Thread1
Thread1 ends
No threads ready or runnable, and no pending interrupts.
Assuming the program completed.
Machine halting!

Ticks: total 30, idle 0, system 30, user 0
Disk I/O: reads 0, writes 0
Console I/O: reads 0, writes 0
Paging: faults 0
Network I/O: packets received 0, sent 0

Cleaning up...
```

Step 5: Save the output to file

```
./nachos > project1_output1.txt
```

This command runs Nachos and saves the output to file project1_output1.txt

Keep the file project1_output1.txt for grading.

Task 2: Adding codes to Nachos

Open file os_nachos_ proj1/threads/project1.cc. All the code you need to write is in this file. There are 4 functions in the file. running_for_calculation(),prior_thread(), latter_thread() and project1(). Your work is to add your code in these 4 functions. Nachos will automatically invoke project1(). Your tasks are:

Step 1:

In Thread1 (i.e. function prior_thread()), invoke function running_for_calculation() to perform certain number of calculations specified by the parameter “arg” . Before the function running_for_calculation() is invoked, output the information which claims that Thread1 will perform calculation for “arg” * 10000 times.

Step 2:

Create second thread called Thread2. This thread invokes function latter_thread () and passes the value 3 to the parameter “arg”.

Step 3:

In Thread2 (i.e. function latter_thread()), invoke function running_for_calculation() to perform certain number of calculations specified by the parameter “arg” . Before the function running_for_calculation() is invoked, output the information which claims that Thread2 will perform calculation for “arg” * 10000 times.

Don't know what to do? Here are some instructions that may be helpful.

1. How to create threads in Nachos?

First, you need to define a Thread object, then invoke Fork(). For example:

```
Thread *th1;
th1 = new Thread("Thread1");
th1->Fork(prior_thread, 2);
```

A thread named "Thread1" will be created, and it invokes prior_thread as its working thread function. The working thread function must have a parameter of int type. e.g. prior_thread() function has a "int arg" parameter. The third line invokes the working thread function and passes the value 2 to the parameter.

Note: Do **NOT** define the object like "Thread th1("Thread1")". This will cause Nachos to crash.

2. How to get the name of a thread in Nachos?

```
currentThread->getName();
```

This function returns the name (pointer of char *) of current thread.

3. About running_for_calculation(int t) function
This function is provided to you. Do NOT modify it. This function will perform the calculation of addition for certain times specified by parameter t.

Step 4:

After you finish coding in Step 3, return to the folder “os_nachos_ proj1” and re-run "make" and "./nachos". Your output should be:

```
Hello, my name is Thread1
Thread1 will perform calculation for 20000 times
Thread1 ends
Hello, my name is Thread2
Thread2 will perform calculation for 30000 times
Thread2 ends
No threads ready or runnable, and no pending interrupts.
Assuming the program completed.
Machine halting!

Ticks: total 50, idle 0, system 50, user 0
Disk I/O: reads 0, writes 0
Console I/O: reads 0, writes 0
Paging: faults 0
Network I/O: packets received 0, sent 0

Cleaning up...
```

Your output is not necessary to be the same but 3 points must be satisfied:
1. Print the name of Thread1 and Thread2.
2. Print the information claiming that thread1 and thread2 is going to perform calculation for certain times with the right parameter.
3. Thread1 ends before Thread2

Step 4: Save the output to file

```
./nachos > project1_output2.txt
```

This command runs the Nachos and save the output to file project1_output2.txt

Keep project1_output2.txt for grading.

Task 3: Simple Thread Scheduling in Nachos

In this task, you are asked to fulfill the following steps.

Step 1: Add one code line in source file project1.cc

At the end of function running_for_calculation(int t) (in line 22), add the code line:

```
currentThread->Yield();
```

What will this Yield() function do?

When the calculation is completed, current thread invokes Yield() function to relinquish the CPU if any other thread is ready to run. If so, current thread is put at the end of the ready list, so that it will then eventually be re-scheduled.

Step 2: Rebuild the project and re-run “nachos” and save the output to file

Enter the project1 folder and run “make” again to rebuild the project.

Right after the previous step, please run command

```
./nachos > project1_output3.txt
```

which saves output of the modified “nachos” to project1_output3.txt.

Please read your output carefully and see if there is any difference between output2.txt and output3.txt.

Step 3: Add several code lines in source file project1.cc

Add **two** code lines to line 40 (within the function latter_thread()) and line 61 (within the function prior_thread()) for printing the global variable “global” information in the two threads and the printed message should claim the thread name where the global variable is. For example, the printed message should be:

```
The global variable in Thread1 is 2
```

Then, please add the code line

```
th1->Suspend();
```

to line 34 of the source file project1.cc (within the function latter_thread()). Please do NOT delete the comment line.

Add the code line

```
th1->Resume();
```

to line 44 of the source file project1.cc (within the function latter_thread()). Please do NOT delete the comment line.

Step 4: Rebuild the project and re-run “nachos”

Enter the project1 folder and run “make” again to rebuild the project. Make sure the building process is without errors or you should revise the code and get rid of them. When you run "./nachos", your output should be:

```
Hello, my name is Thread1
Thread1 will perform calculation for 20000 times
Hello, my name is Thread2
Student's suspend routine called
Thread2 will perform calculation for 30000 times
The global variable in Thread2 is 2
Thread2 ends
The global variable in Thread1 is 3
Thread1 ends
No threads ready or runnable, and no pending interrupts.
Assuming the program completed.
Machine halting!

Ticks: total 90, idle 0, system 90, user 0
Disk I/O: reads 0, writes 0
Console I/O: reads 0, writes 0
Paging: faults 0
Network I/O: packets received 0, sent 0

Cleaning up...
```

Step 5: Save the output to file

Right after the previous step, please run command

```
./nachos > project1_output4.txt
```

which saves output of the modified “nachos” to project1_output4.txt.

Step 6: Scheduling the two threads in a different order

Please **comment out** line 34 and line 44 and **add** two code lines to line 55 and line 65 for scheduling the two threads in a **different order** comparing to Step 3.

Step 7: Rebuild the project and save the output to file

Enter the project1 folder and run “make” again to rebuild the project. Please run command

```
./nachos > project1_output5.txt
```

which saves output of the modified “nachos” to project1_output5.txt.

Step 8: Compare the outputs of each step

Find out the difference between:

(1) The outputs that are saved in project1_output2.txt and project1_output3.txt respectively.

(2) The outputs that are saved in project1_output4.txt and project1_output5.txt respectively.

Describe and explain the difference briefly. You are required to write the answer in project1_report.txt.

Note: Keep project1.cc, project1_output3.txt, project1_output4.txt, project1_output5.txt and project1_report.txt for grading.

After you finish these tasks:

1) Please generate a single file using ZIP and submit it through CASS.

2) The name of the ZIP should be "proj1_********.zip", using your student ID to replace star symbols.

3) The following files should be included insides the ZIP:

File Name	Description
project1.cc	Source file you have accomplished by the end of Task 3
project1_output1.txt	Output of Task 1
project1_output2.txt	Output of Task 2
project1_output3.txt	The first output of Task 3
project1_output4.txt	The second output of Task 3
project1_output5.txt	The third output of Task 3
project1_report.txt	The answers to the question in Step 8 of Task 3