A_6

Assignment VI

Connor Taffe. T no. 3742

March 10th, 2015

2 Report of Lab 4-2

Q. 1 Write the report about your lab task 3(e).

(e) Change the function `SimpleThread()` to bump the shared variable repeatedly as follows:

...

Recompile and run the Nachos again. Observe the printout and final value of `shared`.

First, I set up the lab4-2 directory so I could compile a new, altered, nachos.

```
$ mkdir lab4-2
$ ls
ass3  bin      lab4-2  machine          monitor  threads
ass4  filesys  lab5    Makefile.common  network  userprog
ass5  lab2     lab7-8  Makefile.dep     test     vm
$ cd lab4-2/
$ cp ../threads/threadtest.cc .
$ ls
threadtest.cc
$ cd ../threads/
$ make clean
...
rm -f nachos coff2noff coff2flat
rm -f *.noff *.flat
$ cd ../lab4-2/
$ ls
threadtest.cc
$ cp -r ../threads/arch/ .
$ cp ../threads/Makefile .
$ cp ../threads/Makefile.local .
$ ls
arch  Makefile  Makefile.local  threadtest.cc
```

Then, I edited the source files as specified by part *a* through *e* of Lab 4-2.

```
$ emacs threadtest.cc
...
$ ls
arch  Makefile  Makefile.local  threadtest.cc  threadtest.cc~
$ diff threadtest.cc threadtest.cc~
14d13
< #include "synch.h"
$ ls
arch  Makefile  Makefile.local  threadtest.cc  threadtest.cc~
$ diff threadtest.cc threadtest.cc~
16,18d15
< int shared = 0;
< Semaphore * sem = new Semaphore("Mutex", 1);
<
31,41c28,33
<   int num;
<   for (int i = 0; i < 2; i++) {
<     printf("Thread %d tries to enter critical section %d time.\n", which, i);
<     num =shared;
<     num = num + 1;
<     currentThread->Yield();
<     shared = num;
<     printf("Thread %d is exiting critical section %d time.\n", which, i);
<   }
<   printf("*** Value of Shared is %d, when thread %d is finishing.\n",
<        shared, which);
---
>     int num;
>
>     for (num = 0; num < 5; num++) {
>       printf("*** thread %d looped %d times\n", (int) which, num);
>         currentThread->Yield();
>     }
53c45
<   DEBUG('t', "Entering SimpleTest");
---
>     DEBUG('t', "Entering SimpleTest");
54a47
>     Thread *t = new Thread("forked thread");
56,60c49,50
<   Thread *t;
<   for (int i=0; i<3; i++) {
<     t = new Thread("forked thread");
<     t->Fork(SimpleThread, i);
<   }
---
```

```
>     t->Fork(SimpleThread, 1);
>     SimpleThread(0);
```

Then, I ran `make` to compile a new copy of nachos using the altered source code. And, following that, ran the new `nachos` binary symbolically linked to `nachos` in my current directory.

```
$ make
...
ln -sf arch/unknown-i386-linux/bin/nachos nachos
$ ./nachos
Thread 0 tries to enter critical section 0 time.
Thread 1 tries to enter critical section 0 time.
Thread 2 tries to enter critical section 0 time.
Thread 0 is exiting critical section 0 time.
Thread 0 tries to enter critical section 1 time.
Thread 1 is exiting critical section 0 time.
Thread 1 tries to enter critical section 1 time.
Thread 2 is exiting critical section 0 time.
Thread 2 tries to enter critical section 1 time.
Thread 0 is exiting critical section 1 time.
*** Value of Shared is 2, when thread 0 is finishing.
Thread 1 is exiting critical section 1 time.
*** Value of Shared is 2, when thread 1 is finishing.
Thread 2 is exiting critical section 1 time.
*** Value of Shared is 2, when thread 2 is finishing.
No threads ready or runnable, and no pending interrupts.
Assuming the program completed.
Machine halting!

Ticks: total 130, idle 0, system 130, 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...
```

The final value of `shared` is 2.

Q. 2 Write the report about your lab task 3(f).

(f) Now add semaphore P() and V() calls for the entry and exit codes of the critical sections as follows:

...

Recompile and run the Nachos again. Observe the printout and final value of `shared`.

I edited the `threadtest.cc` to include the calls to P() and V().

```
$ emacs threadtest.cc
...
$ diff threadtest.cc threadtest.cc~
34d33
<     sem->P();
40d38
<     sem->V();
```

Then, I ran `make` to recompile the `nachos` source code with the new P() and V() calls. Afterwards, I ran the new `nachos` with the command `./nachos`.

```
$ make
...
ln -sf arch/unknown-i386-linux/bin/nachos nachos
$ ./nachos
Thread 0 tries to enter critical section 0 time.
Thread 1 tries to enter critical section 0 time.
Thread 2 tries to enter critical section 0 time.
Thread 0 is exiting critical section 0 time.
Thread 0 tries to enter critical section 1 time.
Thread 0 is exiting critical section 1 time.
*** Value of Shared is 2, when thread 0 is finishing.
Thread 2 is exiting critical section 0 time.
Thread 2 tries to enter critical section 1 time.
Thread 2 is exiting critical section 1 time.
*** Value of Shared is 4, when thread 2 is finishing.
Thread 1 is exiting critical section 0 time.
Thread 1 tries to enter critical section 1 time.
Thread 1 is exiting critical section 1 time.
*** Value of Shared is 6, when thread 1 is finishing.
No threads ready or runnable, and no pending interrupts.
Assuming the program completed.
Machine halting!

Ticks: total 250, idle 0, system 250, 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...
```

The final value of `shared` is 6.

3 Questions about Lab 4-2

Q. 1 Mentally trace the execution of this nachos and write (1) the contents of the ready-queue of the systesm, (2) the value of the semaphore sem and (3) the contents of the queue of the semaphore `sem`, when each of the messages above is printed, by filling the table as follows:

Q. 1.1 Thread 0 tries to enter critical section 0 time.

- Ready Queue: head $\longrightarrow t_1 \longrightarrow t_2 \longrightarrow \varnothing$
- Value of `sem`: 1
- Queue of `sem`: head $\longrightarrow \varnothing$

Q. 1.2 Thread 1 tries to enter critical section 0 time.

- Ready Queue: head $\longrightarrow t_2 \longrightarrow t_0 \longrightarrow \varnothing$
- Value of `sem`: 0
- Queue of `sem`: head $\longrightarrow \varnothing$

Q. 1.3 Thread 2 tries to enter critical section 0 time.

- Ready Queue: head $\longrightarrow t_0 \longrightarrow \varnothing$
- Value of `sem`: 0
- Queue of `sem`: head $\longrightarrow t_1 \longrightarrow \varnothing$

Q. 1.4 Thread 0 is exiting critical section 0 time.

- Ready Queue: head $\longrightarrow \varnothing$
- Value of `sem`: 0
- Queue of `sem`: head $\longrightarrow t_1 \longrightarrow t_2 \longrightarrow \varnothing$

Q. 1.5 Thread 0 tries to enter critical section 1 time.

- Ready Queue: head $\longrightarrow t_1 \longrightarrow \varnothing$
- Value of `sem`: 1
- Queue of `sem`: head $\longrightarrow t_2 \longrightarrow \varnothing$

Q. 1.6 Thread 0 is exiting critical section 1 time.

- Ready Queue: head $\longrightarrow \varnothing$
- Value of `sem`: 0

- Queue of `sem`: head $\longrightarrow t_2 \longrightarrow t_1 \longrightarrow \varnothing$

Q. 1.7 *** Value of Shared is 2, when thread 0 is finishing.

- Ready Queue: head $\longrightarrow t_2 \longrightarrow \varnothing$
- Value of `sem`: 1
- Queue of `sem`: head $\longrightarrow t_1 \longrightarrow \varnothing$

Q. 1.8 Thread 2 is exiting critical section 0 time.

- Ready Queue: head $\longrightarrow \varnothing$
- Value of `sem`: 0
- Queue of `sem`: head $\longrightarrow t_1 \longrightarrow \varnothing$

Q. 1.9 Thread 2 tries to enter critical section 1 time.

- Ready Queue: head $\longrightarrow t_1 \longrightarrow \varnothing$
- Value of `sem`: 1
- Queue of `sem`: head $\longrightarrow \varnothing$

Q. 1.10 Thread 2 is exiting critical section 1 time.

- Ready Queue: head $\longrightarrow \varnothing$
- Value of `sem`: 0
- Queue of `sem`: head $\longrightarrow t_1 \longrightarrow \varnothing$

Q. 1.11 *** Value of Shared is 4, when thread 2 is finishing.

- Ready Queue: head $\longrightarrow t_1 \longrightarrow \varnothing$
- Value of `sem`: 1
- Queue of `sem`: head $\longrightarrow \varnothing$

Q. 1.12 Thread 1 is exiting critical section 0 time.

- Ready Queue: head $\longrightarrow \varnothing$
- Value of `sem`: 0
- Queue of `sem`: head $\longrightarrow \varnothing$

Q. 1.13 Thread 1 tries to enter critical section 1 time.

- Ready Queue: head $\longrightarrow \varnothing$
- Value of `sem`: 1

- Queue of `sem`: head $\longrightarrow \varnothing$

Q. 1.14 Thread 1 is exiting critical section 1 time.

- Ready Queue: head $\longrightarrow \varnothing$
- Value of `sem`: 0
- Queue of `sem`: head $\longrightarrow \varnothing$

Q. 1.15 *** Value of Shared is 6, when thread 1 is finishing.

- Ready Queue: head $\longrightarrow \varnothing$
- Value of `sem`: 1
- Queue of `sem`: head $\longrightarrow \varnothing$