Laboratory Quiz 1 – 2021.09.30.09.35.00

Update on 2021.10.07: I scored full mark (10/10) in this laboratory quiz, so all answers that I selected are correct.

Green color highlight denotes my selected answer

Q1: What are the stages of context switch?

1. **Check whether current thread overflowed its stack**
2. **Change the state of new thread to Running**
3. **Perform actual context switch (by calling switch())**
4. **Terminate the previous Thread (if applicable)**

There are other options available, they look similar to the above, but in different orders, and in one of the four options, the 2nd step is not Running but Ready (which is not correct).

Q2: Which function results in the thread state change from running to ready?

- Thread::Fork()
- Thread::Sleep()
- Thread::Finish()
- **Thread::Yield()**

Q3: Which function allocates stack and initializes registers for the thread?

- Yield()
- Finish()
- **Fork()**
- Thread()

Q4: Lab quiz output, then have to fill in the excel sheet

Tick	Ready list	Current thread	Print message	Context switch
90		**Main**		**Main -> child**
100	**Child2, main**		**Thread 1 looped 2 times**	

I didn't record all of the other options available, but in the other options, the Print message(s) are wrong, so it's quite obvious that just by checking the print message you can already determine the correct answer.

Q5: Join flag argument of Thread::Fork() indicates whether or not the invoking parent thread should wait for the newly created child thread to finish its execution

- **True**
- False

During the test I was not too sure about the answer, but after the test, I checked the thread.cc code, I think this is correct.

Q6: In Round Robin, if a thread finishes before the timer interrupt is invoked, then the remaining time for the next thread will be less than 40. Which function and class to modify? (My interpretation: which function and class to invoke the timer reset?)

- Scheduler::Run()
- **Thread::Finish()**
- Thread::Yield()
- Scheduler::ReadyToRun()

Q7: Which is the correct way of calling Thread::Fork() for SimpleTest() by Child1?

- Fork(1, 0, SimpleTest)
- Fork(SimpleTest, 0, 1)
- **Fork(SimpleTest, 1, 0)**

Other answers are Fork(some weird or wrong arguments) so I didn't record

Q8: Which is not true about pending interrupt list in interrupt.cc?

- **This list is sorted in decreasing order of the time ticks when interrupt is triggered**
- Pending timer interrupts are stored in pending interrupt list
- Pending::Schedule inserts new timer interrupts to pending interrupt list
- I forgot the fourth option

I am very certain that the list is sorted in increasing order, also checked the lab manual for lab 2 and verified it's true.

Q9: Which function invokes YieldOnReturn()?

- **TimerInterruptHandler()**
- OneTick()
- I forgot the other options

I am very certain that TimerInterruptHandler() invokes YieldOnReturn(), also checked the lab manual for lab 2 and verified it's true.

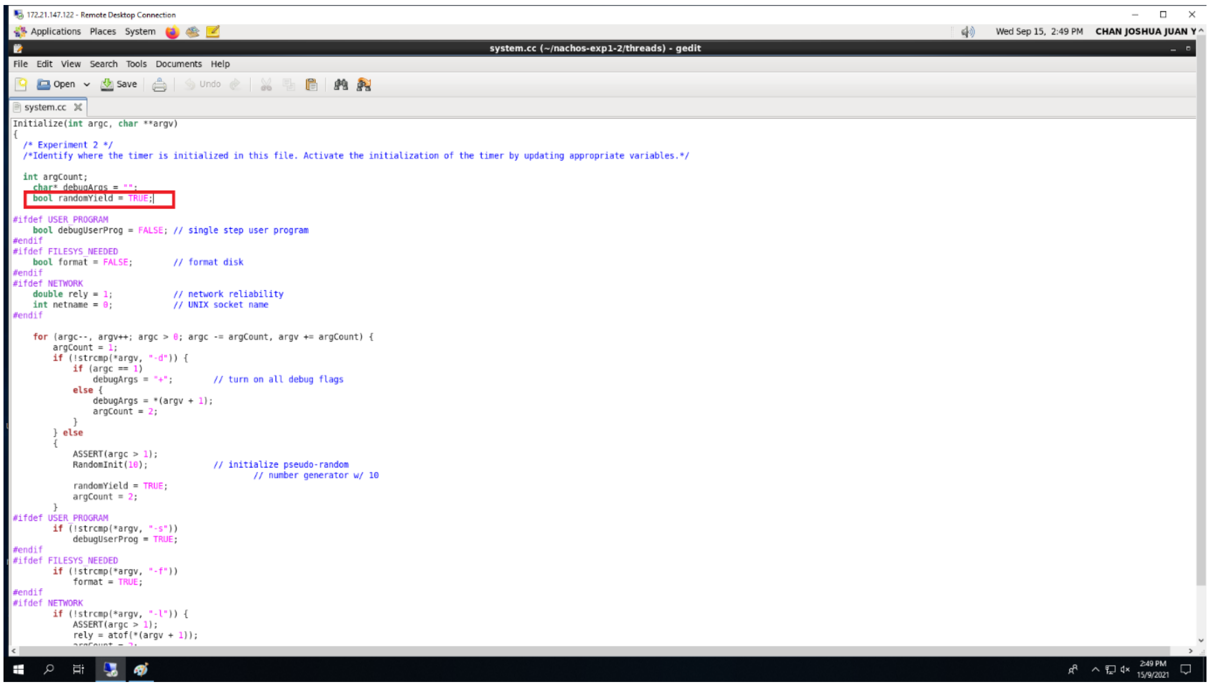
Q10: Below is a section of the the Initialize() function inside system.cc

```
void
Initialize(int argc, char **argv)
{
  /* Experiment 2 */
  /*Identify where the timer is initialized in this file. Activate the
initialization of the timer by updating appropriate variables.*/

  int argCount;
    char* debugArgs = "";
    bool randomYield = TRUE;

#ifdef USER_PROGRAM
    bool debugUserProg = FALSE;     // single step user program
#endif
#ifdef FILESYS_NEEDED
    bool format = FALSE;    // format disk
#endif
#ifdef NETWORK
    double rely = 1;       // network reliability
    int netname = 0;          // UNIX socket name
#endif
```

Determine whether or not the timer will be activated by this call, and which variable is used for this activation?

- No, debugUserProf
- **Yes, randomYield**
- Yes, format
- Yes, debugUserProg

Note: you did this in the lab!!


172.21.147.122 - Remote Desktop Connection
Applications Places System
system.cc (~/nachos-exp1-2/threads) - gedit
File Edit View Search Tools Documents Help
Open Save Undo
system.cc

```
Initialize(int argc, char **argv)
{
  /* Experiment 2 */
  /*Identify where the timer is initialized in this file. Activate the initialization of the timer by updating appropriate variables.*/

  int argCount;
    char* debugArgs = "";
    bool randomYield = TRUE;

#ifdef USER_PROGRAM
    bool debugUserProg = FALSE; // single step user program
#endif
#ifdef FILESYS_NEEDED
    bool format = FALSE;        // format disk
#endif
#ifdef NETWORK
    double rely = 1;            // network reliability
    int netname = 0;            // UNIX socket name
#endif

    for (argc--, argv++; argc > 0; argc -= argCount, argv += argCount) {
        argCount = 1;
        if (!strcmp(*argv, "-d")) {
            if (argc == 1)
                debugArgs = "+";        // turn on all debug flags
            else {
                debugArgs = *(argv + 1);
                argCount = 2;
            }
        } else
        {
            ASSERT(argc > 1);
            RandomInit(10);             // initialize pseudo-random
                                        // number generator w/ 10

            randomYield = TRUE;
            argCount = 2;
        }
#ifdef USER_PROGRAM
        if (!strcmp(*argv, "-s"))
            debugUserProg = TRUE;
#endif
#ifdef FILESYS_NEEDED
        if (!strcmp(*argv, "-f"))
            format = TRUE;
#endif
#ifdef NETWORK
        if (!strcmp(*argv, "-l")) {
            ASSERT(argc > 1);
            rely = atof(*(argv + 1));
```