System and device programming
15 July 2014

(Theory: no textbooks and/or course material allowed)

(15 marks) The final mark is the sum of the 1st > 8 and the 2nd part >10
The final mark cannot be refused, it will be registered (no retry for marks >= 18)

1. (3.0 marks) a) Describe events in the Windows system and their related system calls. Describe the 4 cases of event signal/release, related to manual/auto reset and set/pulse event conditions.
   b) Given a set of threads working in master-slave mode, with one master and N slaves, we want to use events for two purposes:
   - for the master, in order to wait for task completion from a slave (any slave can complete a task under execution at a given time): events are used to communicate completion from a slave to the master
   - for the master, to enable a specific slave to start a new task

   How many events are necessary ? 1, 2, N, N+1, 2N, 2N +1, 2N+2? (motivate your answer)
   Which kind of events (manual/auto, set/pulse) ? Why ?
   (handling of tasks and data is NOT required, just propose a solution for synchronization)

2. (3.0 marks) Describe the main differences between static and dynamic libraries, and answer the following questions:
   - What are implicit and explicit linking, within the framework of dynamic libraries ?
   - When are (explicitly/implicitly linked) libraries linked: at compile- load- or execution-time ?
   - Are DLLs required to be thread-safe (motivate the yes/no answer, and also explain what is thread-safety) ?
   - Can a DLL be shared, at runtime, among processes ?
     - o If (the answer is) yes, can it be used to share data among processes using them (e.g. to hold a shared array for reading/writing) ?
     - o If (the answer is) no, how can a library routine be shared (a single copy is resident in memory) among several processes ?

(next 3 questions in the back)

3. (3.0 marks) Explain the behaviour of WaitForSingleObject and WaitForMultipleObject in WIN32.
   Are calls to the two functions blocking? What can we wait for, with the two functions ? How many, and which, different synchronization schemes are possible through WFMO? Is it possible to use WFMO in order to wait for one among multiple events/objects ?
   Given the two wait loops written below, explain their wait scheme.
   Are the calls to WFMO in loop1/loop2 waiting for completion of all, or for one, among many processes/threads ? If yes, is the waiting order fixed ?
   Is it possible to detect which of the waited processes/threads has completed ?
   What does constant WAIT_OBJECT_0) represent ?
   Why the first parameter in loop1 is min (MAXIMUM_WAIT_OBJECTS, argc - 2 - iProc), instead of nProc ?
   What are hProc and tHandle ?

```
/* loop 1 */
for (iProc = 0; iProc < nProc; iProc += MAXIMUM_WAIT_OBJECTS)
   WaitForMultipleObjects (min (MAXIMUM_WAIT_OBJECTS, argc - 2 - iProc),
                              &hProc[iProc], TRUE, INFINITE);

/* loop 2 */
while (ThdCnt >0)  {
   ThdIdxP = WaitForMultipleObjects (ThdCnt, tHandle, FALSE,INFINITE);
   iThrd = (int) ThdIdxP - (int) WAIT_OBJECT_0;
   if (iThrd < 0 || iThrd >= ThdCnt)
      ReportError (_T (''Thread wait error."), 5, TRUE);
   GetExitCodeThread (tHandle [iThrd], &ExitCode);
   CloseHandle (tHandle [iThrd]);
 }
```

4. (3.0 marks) Detail the steps performed by the system call close, with reference to all the objects and data structures involved by its operations.

5. (3.0 marks) Explain all the arguments of this command line:
   `qemu -hdb fs.img xv6.img -serial mon:stdio -S -gdb tcp::26000 -smp 2 -m 512`

System and device programming
15 July 2014

Examination Time: 1 h 45 min Evaluation: 18 marks.
Textbooks and/or course material allowed.
The final mark is the sum of the 1st and the 2nd parts; it cannot be refused (no retry for marks >= 18).

The binomial coefficient is defined as:

$$\binom{n}{k} = \frac{n!}{(n-k)!\,k!} = \frac{n \cdot (n-1) \cdots\cdots (n-k+1)}{1 \cdot 2 \cdots\cdots k}$$

$$\binom{n}{0} = \binom{n}{n} = 1$$

`n` and `k` are integers, and `n` >= `k` >= 0.

For example $\binom{15}{11} = \frac{15!}{(15-11)!\,11!} = \frac{15 \cdot 14 \cdot 13 \cdot 12 \cdot 11 \cdot 10 \cdot 9 \cdot 8 \cdot 7 \cdot 6 \cdot 5}{1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6 \cdot 7 \cdot 8 \cdot 9 \cdot 10 \cdot 11} = 1365$

Write a concurrent `C` program in the Unix environment which receives two arguments `n`, `k`, and computes and print the binomial coefficient.

The computation has to be performed in concurrency by a **`numerator`** thread a **`denominator:`** thread.

The **`numerator`** thread loops by taking **two factors at a time** that must be multiplied (for example `15` and `14`), computes their product, and save its result in its variable **`numer`**.

The same procedure is performed by the **`denominator`** threads, which save its result in its variable **`denom`**.

After the **numerator** and the **denominator** threads have performed their computation, **they must synchronize: one of them has to perform the division** of the partial contributions stored in **numer** and **denom,** and save it in a global variable **result.**

Then, **`numerator`** and **`denominator`** threads continue their job with the next pairs of factors that have not yet been processed.

Since it is possible that the number of remaining factors in the numerator and in the denominator is less than 2, care has to be taken to synchronize the threads also considering this possibility, so that the synchronization among the **`numerator`** and **`denominator`** threads is correct and each thread can terminate.

The main thread will print the **result** of the binomial coefficient computation.