UM–SJTU Joint Institute

Introduction to Operating Systems
(VE482)

Homework 4

Name: Ji Xingyou ID: 515370910197
Date: 26 October 2017

Contents

1 *Simple questions*

1. **Consider a system in which threads are implemented entirely in user space, with the run-time system getting a clock interrupt once a second. Suppose that a clock interrupt occurs while some thread is executing in the run-time system. What problem might occur? Can you suggest a way to solve it?**

   If the runtime system is blocking a thread and dealing with the scheduling queues at the same time, the clock interrupt handler will not be able to tell whether or not a switch in thread is needed because they might be in an inconsistent state.

   A possible solution is to set a flag for the runtime system and another flag for the clock handler. When the runtime system is finished, it will see the state of the clock flag, thus telling whether a clock interrupt happened or not, then deciding how the clock handler acts.

2. **Suppose that an operating system does not have anything like the select system call (man select for more details on the command) to see in advance if it is safe to read from a file, pipe, or device, but it does allow alarm clocks to be set that interrupt blocked system calls. Is it possible to implement a threads package in user space under these conditions? Discuss.**

   It is possible to implement, but it will be very inefficient to do so. When a thread wants to use a system call, it will first set an alarm timer. If the call blocks, the control is given back to the threads package. Otherwise, the timer will be cleared. Compare to the traditional mode, this is much more inefficient, not to mention other potential problems that may occur if the timer goes off too early.

2 *Monitors*

1. **During the lecture monitors were introduced (3.36). They use condition variables as well as two instructions, wait and signal. A different approach would be to have only one operation called waituntil, which would check the value of a boolean expression and only allow a process to run when it evaluates as True. What would be the drawback of such a solution?**

   Such a solution would be time-consuming. When a process is blocked by the waituntil operation, the system must calculate the boolean expression, which is very likely to take a lot of time since such a boolean expression is always very complicated.

3 *Race condition in Bash*

Write a Bash script which generates a file composed of one integer per line. The script should read the last number in the file, add one to it, and append the result to the file.

1. **Run the script in both background and foreground at the same time. How long does it take before observing a race condition?**

   See the attached ex3_1.sh file.

   The first race comes when two 29 appear.

   See the attached int_1.txt for detail.

2. **Modify the script such as to prevent the race condition.**

   See the attached ex3_2.sh file.

   From the output file, we can see that there is not a race condition this time.

4 *Programming with semaphores*

1. **On Linux, find the file semaphore.h.**

2. **Read the documentation to understand how to use the functions described in the file semaphore.h.**

3. **Using semaphores adjust the program such as to always return the correct answer.**

   See the attached main.c as well as Readme.txt for detail.