VE 482 Homework 4

Ex 1

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 precisely at the point of blocking or unblocking a thread, the threads might be in an inconsistent state, which might cause problems when interrupt handler deals with interrupts.

   To solve it, we can set a flag when the runtime system is entered. When the clock handler see this flag it will set its own flag and return. When the runtime system finished, it would check the clock flag. If it sees that a clock interrupt occurred, the handler will then be run.

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's possible, but it will be much less efficient because extra work is needed to set alarm clocks to threads before an interrupt.

Ex 2

During the lecture monitors were introduced (3.30). 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?

If we use `waituntil`, we need to continuously check the value of the boolean expression to see whether it is `Ture`, which is a waste of resources.

Ex 3

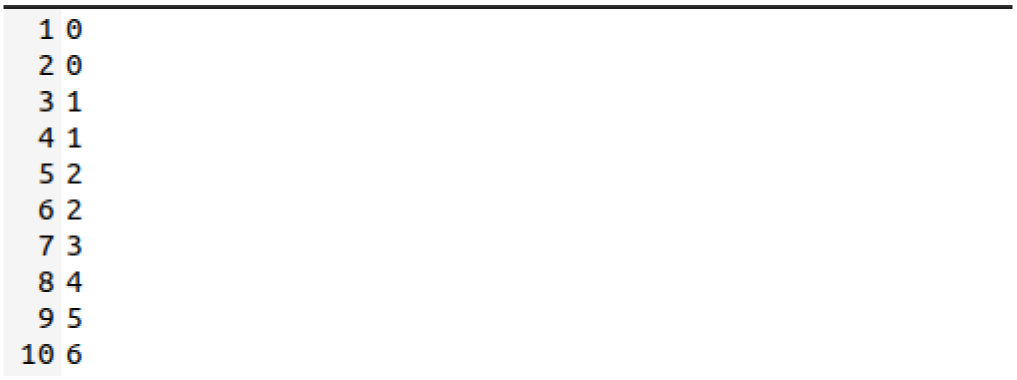
3.1

```bash
#!/bin/bash
if [ ! -f "ex3.txt" ]; then
    echo "0">>ex3.txt
fi
for ((i=0;i<100;i++))
do
    last=$( tail -1 ex3.txt )
    echo $(( last+1 ))>>ex3.txt
done
```

The code is also attached, named as `ex3.sh`.

- **How long does it take before observing a race condition?**

  Race condition occurred from beginning, since even "0" is printed twice. The first ten lines are shown below:



```
0
0
1
1
2
2
3
4
5
6
```

3.2

```bash
#!/bin/bash
if [ ! -f "ex3.txt" ]; then
    (
        flock -e 200
        echo "0">>ex3.txt
    ) 200<>ex3.txt
fi
for ((i=0;i<100;i++))
do
    (
        flock -e 200
        last=$( tail -1 ex3.txt )
        echo $(( last+1 ))>>ex3.txt
    ) 200<>ex3.txt
done
```

The code is also attached, named as `ex3_r.sh`.

Ex 4

See attached file `ex4.c`.