**ECE 2220: System Programming Concepts** **Fall 2020**
**Machine Problem 6: Cyber Defense** **Due: 11:59 pm, Sunday, November 29**

In this lab, each student is to write a **single** program called `lab7.c`. The program utilizes a parent process to create and communicate with child processes. The student specifically explores:

- Handling processes by use of **`fork()`**, **`exit()`**, and **`wait()`**
- Handling signals by use of **`signal()`**, **`alarm()`**, **`pause()`**, and **`kill()`**
- Opening and printing to terminals.

The objective of the program is to model a Cyber Operations center that is monitoring three critical information technology (IT) resources. The system is under attack by Shadow Brokers and the goal is to identify when a hacker is starting to break through defenses. For a hacker to penetrate a system, the hacker first uses network tools to scan for detailed configuration information. Once the hacker has learned all of the network organization, subnetwork design, server locations, port mappings, and firewall locations, they can devise a strategy to exploit weaknesses. However, to gather this information the hacker must inject extensive network traffic that is probing for the configuration details. The Cyber Operations Center monitors for evidence of an attack, and when discovered can initiate a re-configuration of many of the network details (such as renaming IP addresses, moving virtual machines supporting critical services to different hardware or subnetworks, changing subnetwork configurations, modifying firewall settings, and other network administration options). If the Cyber Operations experts reconfigure the network organization and services, then the probing information from the hackers is rendered useless and they have to begin again.

In this game, the Cyber Operations center can perform two operations to each of three IT services. First, the Ops Center can send a SNMP packet requesting a report of log messages from an IT services. However, these messages cause considerable interruption to the ability of the IT services to handle requests, so an SNMP request cannot be sent until after the IT service reports it is able to accept a new command. The second operation by the Ops Center is to send a command to execute a reconfiguration script. This must only be done once the IT service has identified that an attack has reached a critical level. Once the attack is deemed critical, the Ops Center has about a 5 second window to send the command, or the attack succeeds and the IT service is compromised.

The player operates the Ops Center to send SNMP packets to collect the status of the three IT services and to send reconfiguration scripts to interrupt and redeploy the IT services.

## Operation

Before executing the program, the user should open four console (terminal) windows. Type **`tty`** in each terminal window to learn that terminal's **`pts`** number. Then start lab7 and include the four terminal numbers on the command line (e.g., **`./lab7 4 7 9 14`**). The first terminal window represents Cyber Operations center, and the other three represent the three IT services. You can assume that four terminal windows are open when you start your program and that the user lists the correct terminal numbers.

The main function of the parent is to use **`fork()`** to spawn three child processes to model the IT services. There is a child process for each IT service, and each child process executes exactly the same code (except the seed for the pseudo-random number generator). A fourth child is needed to collect input from the Cyber Operations center terminal window and to send signals (via **`kill()`**) to the three IT services. The parent waits for the three IT services to either be protected or crash (the parent collects the exit codes using **`wait()`**). After the parent collects the return values from **`wait()`** it prints a message about the status of the mission for each IT service.

## IT service

Each IT service is to be initialized to a threat level of `1`. A service can receive three signals: (a) a request to print an SNMP report of its status, (b) a command to reconfigure is services, and (c) an alarm that repeats once each second to update its status. Each service should initialize the **`drand48()`** pseudo-random generator using **`srand48()`** and a seed based on time and also unique for each service (see **`snort6.c`** from MP6 for an example for using **`srand48()`**). Configure an initial alarm (`SIGALRM`) using **`alarm(1)`** to set a timer for one second.

The IT service uses **`pause()`** to wait for a signal (one of `SIGALRM`, `SIGUSR1`, or `SIGUSR2`). When the signal is received, check the signal type.

If the signal type indicates that the IT service should be reconfigured then

- Check if a reconfiguration signal has already been received, and if so print a message to the IT window: "**Cannot reconfigure more than once. You are fired!**"
- If the threat level is less than the red level, print a message "**Threat level is not critical. You are fired!**"
- If the player is fired, then exit with the exit code that indicates the IT service crashed.
- Otherwise, mark the IT service as having been reconfigured and print the message: "**Reconfiguring system to thwart attack – this may take a few seconds**"

If the signal type indicates that an SNMP message has been received, then the IT service should

- Check that it has been at least 5 seconds since the last SNMP request. If not, print "**Load too high. Threat is increased**". Then increase the threat level by one.
- Otherwise, print the threat level color (but not the value): **Red** if the level is 10 or more. **Yellow** if the level is 5 or more (but not red). Otherwise, the level is **Green**.

If the signal type indicates an alarm, then the IT service should

- Keep track of the time since the last SNMP request
- Change the threat level. If the system has been reconfigured (via the reconfiguration signal) then decrement the threat level by one. Otherwise, set the new threat level using this function (note the value that is returned is the new threat level):

```
int calc_threat(int threat)
{
    if (drand48() < 0.5) {
        threat++;
    } else if (threat > 1 && drand48() < 0.6) {
        threat--;
    }
    return threat;
}
```

- If it has been less than 5 seconds since the last SNMP status report, then print a message showing the number of seconds until the report can be generated, such as "**Next report available in *x* seconds**". But, if the IT server is permitted to receive a SNMP request, then do not print anything (the SNMP signal must be received to print the threat color).

- If the threat level is above 15, then print "**Intruder! Data stolen …**" and exit with the exit code that indicates the IT service has by compromised.
- If the IT service has been reconfigured and the threat level is now less than that red level, print "**Attack averted. Mission Complete**". Exit the function with a success code.
- Unless the function has exited (due to a crash or successful mitigation of the attack), set a new alarm for one second.

## Input

The program should accept the following input from the terminal associated with the Cyber Operations center (where invalid commands should produce a suitable error message):

- **`sn`** – Sends an SNMP request to IT service **`n`** where **`n`** is one of 1, 2, or 3.
- **`rn`** – Sends a reconfiguration script to IT service **`n`**.
- **`kn`** – Instructs IT service **`n`** to shutdown (the process is killed). **`n`** is 1, 2, or 3.

If the input is one of the **`kn`** commands, the Monitor should terminate the selected child process and print "**`Terminated IT service n`**" where **`n`** is the IT service number.

Note that the parent process must fork a child process to handle the Monitor commands from the keyboard. The parent process uses the **`wait()`** command to collect the exit codes from the three IT services. Once the parent has collected the status of all three IT services, it must print a message such as below to describe the outcome of each IT service:

- "**`IT service`** ***n*** **`compromised, we are going out of business!`**"
- "**`Call HR, we need a new cybersecurity expert for service`** *n*"
- "**`Job well done IT specialist`** ***n*****`. Prepare for new attacks!`**"

Beware that if the parent process terminates and there is a child process waiting for keyboard input, then the child process will spin out of control and be stuck in an infinite loop. So, be sure all child processes have terminated.

## Notes

1. Examples of **all** the techniques needed for this machine problem are found in the lecture notes. See the **`wait.c`**, **`kill.c`**, **`alarm.c`**, and **`signal.c`** series of files. Do **not** use the **`sleep`** function as it interferes with the **`alarm`** function.

2. Use **`ps -ef | grep lab7`** to verify that all of your child processes have terminated. Your final program must not leave any processes hanging. During debugging, if you find a process that is hung, use **`kill pid`** to end the process, where **`pid`** is the process number.

3. Environment comments: Older versions of VirtualBox do not correctly implement alarms, so if you normally use VirtualBox you must ssh to another machine such as the CES apollo computers. VMware and newer versions of VitrualBox work correctly.

4. Submission requirements: The code you submit must compile using the `-Wall` flag and **no** compiler errors or warnings are permitted. To receive credit for this assignment your code must

correctly print to multiple terminals and demonstrate handing of processes and signals. Code that does not compile or fails to pass the minimum tests will not be accepted or graded.

5. Use **`openterms.c`** for reference.

6. Upload your **`lab7.c`** to Canvas.

See the ECE 2220 Programming Guide for additional requirements that apply to all programming assignments.

Work must be completed by each individual student. See the course syllabus for additional policies.