Program Report

KV5002 Computer Networks, Security, and Operating Systems

Krystof Hruby (W20021023)

# 1. The Lunar Lander Controller

## A. Threads and Semaphores

Threads allow a program to split its processes into different parts, which are executed concurrently and separately. The program uses five threads – one for the display, one for the user keyboard input, one for lander communication, one for the dashboard and one for logging data. The program uses threads, since it is not meant to be executed consecutively, for example while it waits for the user to press a key, it must still process and display the game and communicate with the server and log data.

Since the threads of the program must share data with each other (for example the input key, received in the keyboard input managing thread, must be available for the lander managing thread to be processed and communicated with the server, or for the display thread to be shown on the screen). The communication is done through global (accessible from every thread) variables, such as “landercommand” or “landerstate”. Since the program does not have a (full) control of when is each thread being executed, this may lead to interference – two threads modifying the same variable at the same time, thus causing unwanted behaviour. This is also known as a race condition.

To prevent this, the code uses so called semaphores, a mechanism first proposed by Edsger Dijkstra in 1962. Each thread may have a critical section(s), which, once reached, must be executed consecutively, thus blocking other threads to enter their critical section(s) until it is finished. This is managed by special variables, called semaphores (initialized in main), with two functions – “sem\_wait” – queue the thread and once it is its turn start executing the critical section, and “sem\_post” – exit a critical section and unlock it to other threads. These sections are used in keyboard managing, display managing, and lander managing (implicitly through “parsecondition” and “parsestate” functions) threads. The execution of threads and the semaphore queue are controlled by the operating system.

Semaphores solve the problem of synchronization between threads. Compared to other options, they also avoid “busy wait” (repeatedly checking for a condition to be met), since the processes are queued, and thus avoid a larger overhead. The disadvantage of semaphores (and of using a queue system) is that low priority processes may block high priority processes in the execution order, this is called a priority inversion. A different approach for the program might be for example through the Peterson’s Algorithm for Mutual Lock (1981), however this mechanism does not avoid busy wait and may bring larger overhead costs compared to a semaphore-based solution.

User input is managed in the “display” function which runs as its own thread. It waits for the user to press a key (arrow key up, down, left, or right). Once an input is received, it enters its critical section by calling sem\_wait, where it parses the input and modifies the global variables (specifically the lander’s thrust and rotation) based on the input. It calls sem\_post afterwards, exiting its critical section.

## B. UDP and TCP

## C. Data Logging

Data logging is a process of collecting and storing data over a period of time. The program logs data each 5 seconds into a csv (comma-separated values) file named “log.csv”. Data logged is formatted in a JSON (JavaScript Object Notation) format (example of several records logged into the file are shown in Appendix 4).

The program logs several different values. Each data point is a “time:state” value pair, where “time” represents the time of when the data point was logged and “state” the state of the game at that time. Logging time will help to put the logs into the order in which operations were performed and thus create an overall picture of the execution of operations from start to finish.

The “state” part of a data point consists of user input (key pressed by the user) and the state of lander, which is further divided into “command” (“thrust” and “rotation”), “state” (“x”, “y”, “O”, “dx”, “dy”, “dO”) and “condition” (“fuel”, “altitude”, “contact”). “Key” can be of values “up”, “down”, “left” and “right”; all other data is of value float, besides “contact”, which is a boolean (true of false). Logging all this data allows for a recreation of the exact state the lander was in at any point during the execution of the program, which in combination with the logged time allows for a full reenactment of the game from beginning to end.

The data is logged every 5 seconds. The interval in which data should be logged depends on the particular system it is logging the data for and the use case for the logged data. A balance must be found between a small interval, which captures the dynamics of the environment in detail, and a large interval, which lowers the impact on performance and the amount of storage needed for the logs. The data to be logged must be also correctly picked, such that important data is recorded, while redundant information omitted. The size of available storage for the logs and the method of logging may also play a significant role in what data should be logged and how often it should be logged. In some situations, data is not logged periodically over time, but instead tied to a completion of particular events (for example when the game is started or when the player presses a key).

For the Lunar Lander Controller, in my opinion, the data could be logged in smaller intervals (than every 5 seconds), since one game takes a rather short amount of time to complete. This makes it produce only a few records, which do not give a proper, full overview of the game. Logging in even larger intervals could potentially make it produce only a single record per game (or per several games), making logging ineffective or even completely meaningless. The amount and information logged would also depend on the actual application of the data.

The size of the log file grows linearly over time – one record per 5 seconds. How much of space does a new record take compared to the whole log file can be calculated as . One log has a size varying between 232 and 271 characters with one additional character (comma) as a delimiter, thus, the size of the file will grow on average (mean) by 252.5 characters per 5 seconds. One character in C takes one byte of memory, thus, the file will expand on average by 2,020 bits per 5 seconds (3,030 bytes per minute; 181.8 kilobytes per hour). Graphs comparing different rates of the log file growth can be found in Appendix 5.

(a) A discussion of the threads and semaphores you used within your code. What could be used in code as an alternative to semaphores and threads? What are the advantages and disadvantages of using them in fulfilling the functional requirements of the program? How has the program managed user input (6 marks)

(b) Instead of UDP, write another code, in the report, related to communication between controller and the server via TCP, assuming that server listens to TCP packets (the communications protocol is described in detail in Section 1.3.1). What could be the advantages and disadvantages of using UDP instead of TCP for the lunar lander code? (7 marks)

(c) A description of the data logging that you have planned. What data should the program log and why? What could be the advantages and disadvantages of logging data five times per second? How often would you be logging data if you weren't assigned a fixed interval and why? At what rate does the size of the file grow while the lander is being controlled? How have you determined this? (7 marks)

You have to include the following three parts for Network and Operating Systems Programming (50%) in your submission:

(1) Please compress all your codes to an archive file (such as a zip file) and upload it to the blackboard Assessment -> Program. More details can be found in Section 1.6.

(2) You must include your codes as appendix 1 in your report. More details can be found in Section 1.6.

(3) The first section entitled ‘The lunar lander controller’. More details can be found in Section 1.8.

# 2. Operating Systems Theory and Concepts

## A. Advantages of a Command Line in Linux OS

The main advantage of using a command line is its greater speed of performing many (in particular complex) actions, some of which cannot be even achieved through a graphical user interface (GUI). On top of that, additional functionality can be added through scripting. The user can for example quickly move through the computer’s filesystem, compile and run code, edit (text-based) documents or even create short applications to automate tasks.

Another advantage of the command line is that it is lightweight - it uses less processing power and memory compared to a GUI. This makes it perfect in situations in which such things are limited, such as server-side programming, or in less-powerful machines.

The third advantage is that a command line can be used even when GUI is not present or not necessary (for example, in hardware-related or server-side applications).

## B. Processes and Threads

Process (sometimes referred to as “task”) is an instance of a program scheduled for execution by the operating system. It includes all the necessary information needed for running that particular program, such as the process ID, the program’s code and its environment variables, virtual address space and its size. A process can be in several different states depending on the execution circumstance it is currently in – new, ready, running, blocked (waiting for interrupt or other process), terminated. A process can create child processes; they can either share all, some, or no resources. The parent may wait for its child processes to finish execution, or they may execute simultaneously. A process is isolated from other processes – it does not share memory with other processes. The creation, termination and execution of processes are usually handled by the operating system itself.

Thread is a segment of a process. All threads belonging to a process share common system resources and each thread has a subset of its process’s address space. A process may have one or multiple threads. Threads of a process may share some resources, however since they may run in parallel, such resources must be carefully managed to ensure a proper execution of the program. The creation, termination and execution of threads is usually handled by their corresponding process.

Scenario 1: Use of multiple threads (multiple windows of a web browser)

Scenario 2: Use of multiple processes (text editor and a web browser)

## C. Context Switching in an Operating System

Since most machines use only a single CPU, but still need to run several processes at the same time, the operating system must perform a context switch, such that the CPU may be used by multiple processes. A context switch is also used between threads.

Context switch simply means changing from one currently executed process to another. During a context switch, the old process’s state must be saved to the Process Control Block (PCB) (this will ensure that once resumed, the paused process will start executing from the same point at which it ended) and its state updated to blocked, new process is then selected by the scheduler for execution and loaded from the PCB into the memory, its state is changed to running and it starts being executed. [Shown in Figure 1 below]

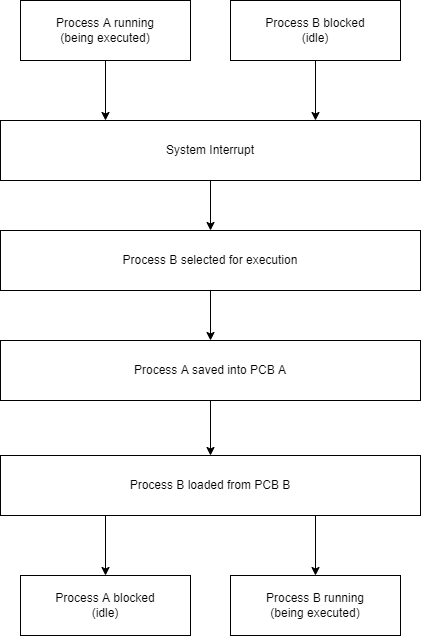
A context switch may occur under different circumstances, these may be for example a system interrupt (the current program must wait for input/output or a server response), or a timer interrupt (at regular intervals, the operating system automatically switches between one process to another). A context switch is considered to be an expensive operation, since it uses the system’s resources, while not performing any actual task. There are several overhead costs – saving, restoring and scheduling processes and the loss of cached resources.

Figure , This figure shows a diagram of a context switch between two processes

Ubuntu is a popular Linux operating system. Ubuntu has server versions and desktops versions. Server versions of Ubuntu are headless, which means users can only use terminals to operate computers in which a server version of Ubuntu is installed. For example, you can use Nano or Vim to edit documents, and use gcc command to compile your c code. Additional to the terminal based interfaces, desktop versions of Ubuntu have Graphical User Interfaces (GUI), which is more user friendly in some ways. For example, you can use a Chrome Web browser with multiple windows. You can also use text editors (such as LibreOffice Writer, an MS word document editor in Ubuntu) when you use the Chrome web browser.

(a) Please list three advantages of command line in Linux Operating Systems and give one example for each advantage. (9 marks)

(b) Please describe the difference between the processes and threads (6 marks). In the introduction of application scenario of Ubuntu, the following two scenarios are mentioned: [Scenario 1]: you can use a Chrome Web browser with multiple windows; [Scenario 2]: You can also use text editors (such as LibreOffice Writer, an MS word document editor in Ubuntu) when you use the Chrome web browser. Which scenario is the use of multiple threads (1 mark)? Which scenario is the use of multiple processes (1 mark)?

(c) Context switch happens between processes in Ubuntu. Describe in detail the actions taken by an operating system to achieve a context switch between processes. Illustrate your answer with diagrams. Please draw diagrams according to your own understanding and do not copy them from other sources. (8 marks)

# 3. Security

## A. Secure Socket Layer Protection

Processes in an operating system are usually independent of each other – one process does not affect other processes, however in some cases the processes must have a way of communicating. This may be done either through a shared portion of memory which can be accessed and modified by multiple processes or through passing messages between the processes. Inter-process communication is commonly used for server-client communication – the client requests a response and the server answers it. Since such communication is usually carried over the internet and may include sensitive data, the communication is secured by using an SSL protocol.

The browser will connect the user to a server through https – secure hypertext transfer protocol, which uses cryptographic methods such as prime factorization and private/public keys to encrypt the data that is sent between the client and the server over the internet. This means that even if the data was compromised, the attacker would not be able to decrypt it and use it.

SSL uses a method called “three-way handshake” to ensure secure connection. The process is further described below.

## B. Symmetric (Private-Key) and Asymmetric (Public-Key) Cryptography

When some data is encrypted, it is impossible to make sense of it without decrypting it – a specific decryption “key” is needed.

In symmetric cryptography, the same key is used for both encryption and decryption of data. If Alice wants to securely send Bob a file, she first encrypts its contents with a chosen key, sends the encrypted file to Bob, and he then decrypts it with the same key. The main advantage of symmetric cryptography over asymmetric cryptography is speed, however, since both parties need to have access to the same key, it needs to be securely transmitted and safely stored.

In asymmetric cryptography, two keys – public and private – are used instead. Public keys are shared and can be known by anybody, while private keys are only known to a single party; only the combination of both keys allows one to decrypt the encrypted data. If Alice wants to securely send Bob a file, she encrypts its contents with his public key (can be known by anyone), sends the encrypted file to him and he then decrypts it with his private key, which is only known to him. Asymmetric cryptography is slower than symmetric cryptography, however no private keys need to be shared and if one is stolen, since both parties use separate private keys, only one party’s security is compromised.

In SSL, both methods are used for secure communication. The process of establishing a secure connection is called “three-way handshake”. The client (browser) sends a request to the server containing its available cipher suites and SLL versions, the server replies with its certificate, signed by a trusted authority (i.e., Google). After the client verifies this certificate (using asymmetric cryptography), the client and the server then generate a private master key which they use for encrypting and decrypting data. Asymmetric cryptography is used as a secure channel for exchanging this master key and symmetric cryptography using this key is then used for faster communication.

## C. Distributed Denial of service (DDoS) Attack

In distributed denial of service attack (DDoS), the target (server) is flooded with fraudulent incoming traffic from multiple sources (distributed), thus depleting the server’s resources (CPU, memory, bandwidth), which makes it unable to operate properly for legitimate users (denial of service). Sometimes a large number of computers are infected with a virus and then unknowingly used in a DDoS attack – this is usually referred to as “botnet”.

SSL is particularly susceptible to DDoS attack since it is computationally expensive and must be performed for each client connecting to the server (to establish secure connection). After a secure connection is established, the attacker client can immediately request a renegotiation of the contract again and again, or, if renegotiating is disabled by the server, the client immediately closes the connection and opens a new one, triggering the SSL protocol again. This way, the server becomes stuck establishing meaningless connections with fraudulent clients and cannot serve its legitimate users. [Shown if Figure 2 below]

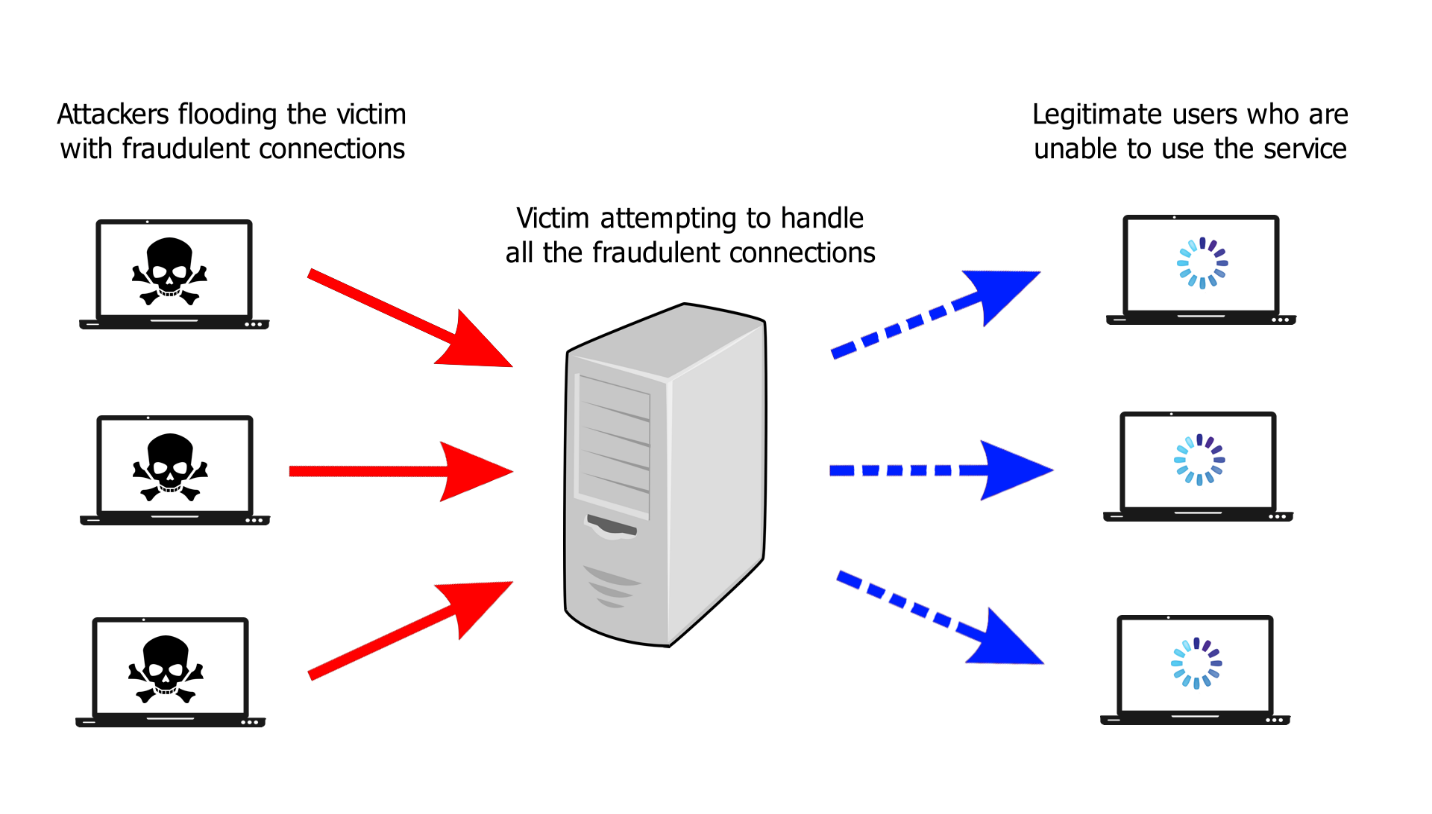
DDoS is known to be a hard to prevent attack, since the server would have to analyze each individual connection and decide which are legitimate and which are fraudulent, which would take a lot of resources. On top of that, if fraudulent requests are coming from a large number of sources, the sever cannot keep up with detecting and closing all such connections. The server may protect itself by monitoring incoming traffic or by employing specific protection applications or services.

Figure 2, This figure shows the nature of a DDoS attack

An important requirement of many operating systems is to provide a secure communication function between processes. One approach to the provision of such a function is the Secure Socket Layer (SSL).

(a) Explain in detail how SSL offers protection for operating systems both for inter-processes and also intra-processes. Also explain how SSL can be used for server to browser transactions. (8 marks)

(b) SSL makes use of both symmetric and public-key cryptography. Explain these concepts (i.e. symmetric cryptography and public-key cryptography), distinguishing clearly between them. Give examples of applications of each cryptography method to show how each of these cryptographic techniques is used in SSL, explaining the reasons for the choice of technique in each case. (8 marks)

(c) Explain if SSL is susceptible to DDOS attacks. Explain the nature of this attack. Illustrate your answer with one diagram. Identify the precise vulnerability of SSL to this attack and discuss how users of SSL can protect themselves against it. Please draw diagrams according to your own understanding and do not copy them from other sources. (9 marks)

# References

Dijkstra, E. W. (1968) “Cooperating Sequential Processes”, in Hansen, P.B. (eds) *The Origin of Concurrent Programming*. Springer, New York, NY. <https://doi.org/10.1007/978-1-4757-3472-0_2>

Peterson, G. L. (1981) “Myths About the Mutual Exclusion Problem”, *Information Processing Letters* 12(3), 115–116

# Appendix

## Appendix 1 (File: controller.c)

## Appendix 2 (File: libnet.c)

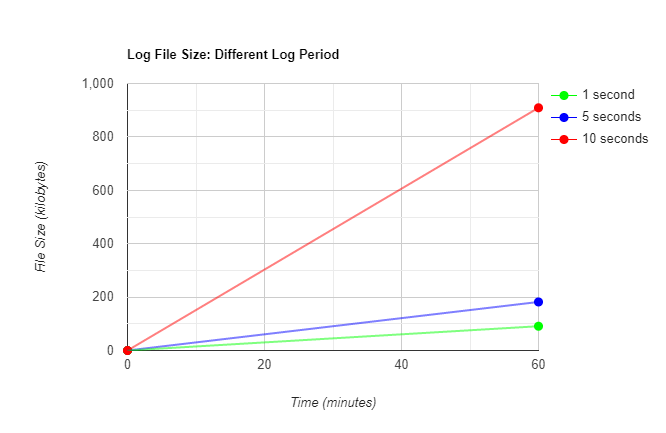
## Appendix 3 (TCP Version of the Communication Between the Controller and the Server)

## Appendix 4 (File: log.csv)

## Appendix 5 (Log File Expansion Rate)

Chart, line chart

Description automatically generatedThe first graph shows how the differing lengths of the record message affect the size of the file over time. Average = 252.5 characters; Shortest = 233 characters; Longest = 272 characters.

The second graph shows how would different periods of logging the data affect the file size over time.