KV5002: Computer networks, security, and operating systems
Course Work 2018-19

Module tutor: David Kendall

1 Dates and mechanisms for assessment submission and feedback

Date of hand out to students: 4 March 2019

Mechanism to be used to disseminate to students: eLP

Date and time of submission by student:

- Report and C program: 23.59 on 16 May 2019

Mechanism for submission of work by student:

- The report should be submitted via eLP using the Turnitin link `Assessment->Report`.
- The C program `controller.c` should be submittted via eLP using the assignment link `Asessement->Program`

Date by which work, feedback and marks will be returned to students: 13 June 2019

Mechanism(s) for return of assignment work, feedback and marks to students: email and appointment on request

2 Network and Operating Systems Programming (50%)

2.1 Overview

In the classic Lunar lander games the player controls the descent of a Lunar Lander spacecraft onto the surface of the moon. [1] [2]

You are to write a program to control a lunar lander. You will be provided with a server that models the flight dynamics and propulsion of the lander. Your program should communicate with the server via UDP, using the application layer protocol described in section 2.3.1. The purpose of communication with the server is to control the flight of the lander. Your program should also communicate with a dashboard display to show the current status of the lander. Finally, the program should write to a data logging file to record the inputs from the user and the response of the lander. Figure 1 shows an overview of the system.

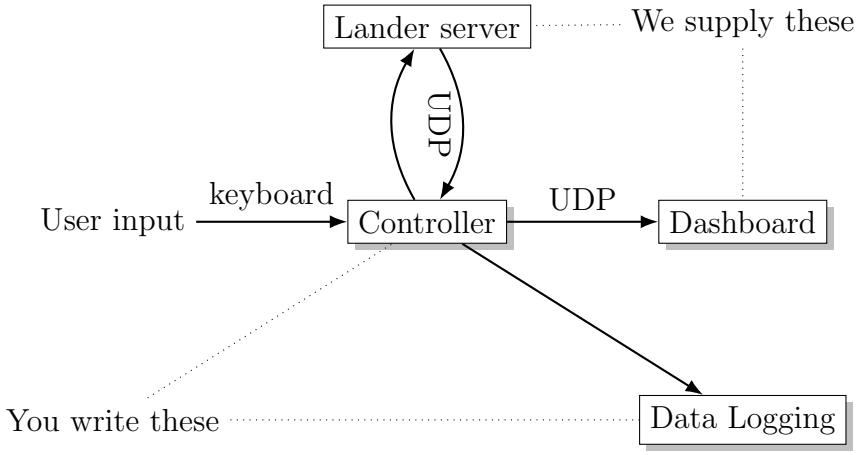

Figure 1: Architecture of the lunar lander software

[1] `https://en.wikipedia.org/wiki/Lunar_Lander_(video_game_genre)`
[2] `https://en.wikipedia.org/wiki/Lunar_Lander_(1979_video_game)`

2.2 Controller

2.2.1 Inputs

Using the keyboard of your PC, accept inputs to control the throttle and the strength of the rotational thrust of the lander. You may choose the mapping of input keys to control signals for yourself. This should be documented in your report.

2.2.2 Communications

The controller needs to manage its communications with the Lander server and the dashboard. Communications with the Lander server are by UDP Datagrams, the format of the messages and responses are given in section 2.3.1. Communications with the dashboard should also use UDP. You should design and implement your own protocol for this purpose. This should be documented in your report.

2.2.3 Control Logic

The controller needs to take the requested actions from the user, and the current state of the lander. From these it should generate the control signals to send to the lander server and the outputs presented to the dashboard. It should also log data to a text file. You should organise your program using at least 4 POSIX threads: user input, server communications, dashboard communications, and data loggging. Any data shared by these threads should be protected, as required, by the use of semaphores.

2.3 Lander Server

The lander server is a Java program that models the dynamics of the lander and the lunar surface. You should clone the repository at https://github.com/dr-alun-moon/LunarLander and follow the instructions to build the program. The lander server is packaged as a `jar` file and executed using

```
java -jar LunarLander.jar
```

2.3.1 Communications Protocol

The server listens for UDP Packets.

Messages to the server, and replies from the server, are formatted as Key:Value pairs, as is done in email headers (RFC822[3]) or HTTP Headers (RFC7230[4]).

Query messages have the message name as the first key, with a single question mark (ascii 63) as the value.

Command messages that set values or cause actions, have the message name as the first key and a corresponding exclamation mark (ascii 33) as the value.

Reply messages, returned in response to the above, have the message name as the key and an equals sign (ascii 61) as the value.

Request State This message requests the current state of the Lander:

Message send

```
state:?
```

Reply from server

```
state:=
x:45.6
y:10.3
O:5
x':0.1
y':-1
O':+0.01
```

The keys and values are:

`x`	x position
`y`	y position
`O`	θ orientation (capital O)
`x'`	$\dot{x}$ horizontal velocity (x single-quote(ascii 39))
`y'`	$\dot{y}$ vertical velocity (y single-quote(ascii 39))
`O'`	$\dot{\theta}$ rate of rotation (capital O single-quote(ascii 39)))

[3] `https://tools.ietf.org/html/rfc822`
[4] `https://tools.ietf.org/html/rfc7230`

Request condition This message requests the general condition of the Lander.

Message send

```
condition:?
```

Reply from server

```
condition:=
fuel:98%
altitude: 20.7
contact:flying|down|crashed
```

The keys and values are:

`fuel`	percentage of fuel remaining	
`altitude`	the radar altitude above the ground	
`contact`	A keyword describing the general state of the Lander	
	`flying`	moving above the terrain
	`down`	in contact with the ground in a successful landing
	`crashed`	in contact or underground in a failed landing

Command Engines This message sets the requested levels on the engines

Message send

```
command:!
main-engine: 100
rcs-roll:  +0.5
```

The keys and values are:

`main-engine`	percentage throttle setting	0...100
`rcs-roll`	the strength of the rotational thrust +ve is anti-clockwise	-1...1,

Either key-value pair may be used as a single item in the message, or both used in the same message.

Reply from server

```
command:=
```

Request Terrain This message requests the terrain below the lander from its mapping radar

Message send

```
terrain:?
```

Reply from server

```
tarrain:=
points:10
data-x: 10.0,15.0,20.0,...
data-y: 12.4,12.7,13,14.0,...
```

The keys and values are:

`points` The number of data points

`data-x` an array of x ground coordinates

`data-y` an array of y ground coordinates

The ground coordinates are supplied one (x, y) pair per line of text. The coordinates are absolute and measured in the same system as the lander state.

> If the Lander's (x, y) position matches a ground (x, y) position, it has reached that point on the ground.
>
> The altitude reported should be the difference between y coordinates of the lander and the ground directly below it.

2.4 Game Dashboard

You are provided with a limited interface on the server, mainly for monitoring the server and debugging. The dashboard provides a more detailed user interface than can be provided by the server.

2.4.1 Requirements

The Dashboard should run on a PC or other computer connected to the controller via a network. You should clone the repository at https://github.com/dr-alun-moon/LanderDash and follow the instructions to build the program. The dashboard can be executed using the command:

```
java -jar LanderDash.jar
```

The dashboard accepts UDP datagrams as input. You can modify the communications class of the dashboard to implement your own protocol. You will need to use a protocol that describes the format of the data as sent in the UDP datagrams. As the dashboard is written in Java and your controller is written in C, there is a danger of miscommunicating. A protocol allows you to describe the format of the data, independently from the programming languages, and you'll have a better chance of success. You should send messages to the dashboard to maintain the current status of all dashboard controls and your protocol should be able to support this requirement.

2.5 Additional requirements

You must include your code for `controller.c` as appendix 1 in your report. It must be possible also to download the raw code file (`controller.c`) from the eLP at `Assessment -> Program` and to build and run the controller with the lander server and dashboard, exactly as described earlier. Failure to satisfy one or more of these requirements will lead automatically to the award of a mark of 0 for the program as described in section 2.6

2.6 Marking of the program

Your program will be assessed on:

1. the extent to which it satisfies the specified functional requirements
   (a) interaction with the user via the keyboard *(2 marks)*
   (b) communication with, and control of, the lander *(4 marks)*
   (c) updating of the dashboard *(4 marks)*
   (d) data logging *(2 marks)*
2. the quality of the code: correct use of threads, semaphores, and network APIs; functional decomposition, data structures, layout, naming etc. Note, when built with `gcc -Wall`, your program should compile without errors or warnings. *(12 marks)*

2.7 What to include about the program in the report

The first section of your report should be entitled 'The lunar lander controller' and should include the following:

1. *The lunar lander controller*

   (a) A discussion of the software architecture of your program, focusing in particular on the use of threads and semaphores. What are the advantages and disadvantages of this architecture in fulfilling the functional requirements of the program? How have you managed user input?

   (6 marks)

   (b) A description and critical evaluation of the protocol that you have used to communicate with the dashboard. To what extent does your protocol help to ensure that communication is reliable?

   (5 marks)

   (c) A description of the data logging that you have performed. What data have you logged and why? What type of file did you use to store the data? Why did you choose this type? How often do you log data? Justify this decision. At what rate does the size of the file grow while the lander is being controlled? How have you determined this?

   (5 marks)

3 OS theory and concepts (50%)

Sections 2 and 3 of your report should include the following:

2. *OS abstraction and process management*

   (a) It is commonly observed that an operating system acts both as a *hardware abstraction layer* and as a *resource manager*. Explain what is meant be each of these terms and give *two* examples of an OS acting in each of these ways.

   (10 marks)

(b) Describe in detail the actions taken by an operating system to achieve a context switch between processes. Illustrate your answer with diagrams.

(10 marks)

3. *Security*
   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) Identify what potential security violations are addressed by SSL. Explain briefly how SSL offers protection against each potential violation that you identify.

   (6 marks)

   (b) SSL makes use of both *symmetric* and *public-key* cryptography. Explain what you understand by these concepts, distinguishing clearly between them. Give examples of each. Identify how each of these cryptographic techniques is used in SSL, explaining the reasons for the choice of technique in each case.

   (7 marks)

   (c) SSL is susceptible to a *man-in-the-middle* attack. Explain the nature of this attack. Identify the precise vulnerability of SSL to this attack and discuss how users of SSL can protect themselves against it.

   (7 marks)

4 Further information

Learning Outcomes assessed in this assessment:

1. Demonstrate knowledge and critical understanding of networking fundamentals, including network architecture and protocols.
2. Demonstrate knowledge and critical understanding of the fundamentals of an operating system, including its architecture and the implementation of its services.
3. Demonstrate knowledge and critical understanding of operating systems and network security, including fundamental concepts, threats, attacks and basic techniques for defence.

4. Interpret a requirements specification to design and develop a networked application program, using standard operating systems and networking APIs (e.g. POSIX.1-2008), and demonstrating cognisance of security issues and industry best practice.
5. Communicate the results of work/study reliably and accurately.

Assessment Criteria/Mark Scheme: The coursework consists of

1. a software development project assessing practical understanding of operating systems, concurrency, and networks (50%)
2. a report assessing understanding of principles and concepts of networks, security, and operating systems (50%)

More detailed marks allocation is provided in section 2 and section 3.

Referencing Style: Harvard

Expected size of the submission: Your report should be about 5 to 7 A4 pages in length (assuming 10pt, normal margins, and excluding appendices). There is no fixed penalty for exceeding this limit but unnecessary verbosity, irrelevance and 'padding' make it difficult for the marker to identify relevant material and may lead to some loss of marks.

Assignment weighting: 100%

Academic Integrity Statement: You must adhere to the university regulations on academic conduct. Formal inquiry proceedings will be instigated if there is any suspicion of plagiarism or any other form of misconduct in your work. Refer to the University's Assessment Regulations for Northumbria Awards if you are unclear as to the meaning of these terms. The latest copy is available on the University website.