COMP 205 - Systems Programming C Shell Group Project Report

Nur IMECE – Dilay SAPMAZ – Ibrahim Ata MERCAN

Abstract

This report intends to intensify the shell programming and c programming languages and aims to make various projects that affect the system.

Keywords: systems programming, c programming, shell programming, UNIX, Linux.

1. Introduction

Various things will be used in this project. Some of these are C programming, Shell programming, and Linux operating system.

The C programming language is an intermediate language. In other words, low-level languages are languages that require a lot of code writing close to the machine language, and high-level languages are languages that require less code writing but run a lot of code in the background. The C programming language is between these two levels.

For Windows, Linux and Unix, the program can be written using the C programming language. In addition, the C programming language has been developed with Assembly language.

A shell is a software that examines the commands entered and finds the necessary resources to pass them to the kernel. Finds the resources required by the commands and pass them to the resources to use. Commands are instructions we send to the system as we know and use them. Delete, copy and so on. There are operating system-specific commands that enable us to perform operations.

2. Main Programming Terms and Contents Used

2.1. What is Systems Programming?

Systems programming corresponds to the software development model in which developers communicate with the system through a lower-level layer so that improvements are made closer to the system. Languages used to develop system programs tend to be down-level. A certain amount of theory and engineering knowledge is required to write them. System programming is the heavy industry of software. Therefore, it is a bit troubling to find people working in this field in our country and trying to improve themselves.

Today, system programming is practically practiced through lower-level languages, rather than developing applications in higher-level languages. The most commonly used languages for system programming are C and C ++. It requires communication with the operating system kernel through functions and system calls in the standard C library. The fact that system programming is done in lower-level languages does not mean that the experience gained in this issue will not work in higher-level languages. On the contrary, knowing the lower level has a reinforcing effect. However, when working in a high-level language, system programming parts are often invisible or difficult to see for us.

The system programmer is simply the person who writes system programs such as operating systems, device drivers. System programming is the most difficult but most respected field of programming. The system programmer is always close to the machine. It is in direct communication with the hardware. Manages the most basic system resources.

Today, operating systems and drivers are written in approximately 80% C language. The remaining part is written in machine language. C is a very strong and flexible language due to its structure and purpose of creation. The purpose of C is to be an alternative to machine language and is designed entirely for system programming purposes.

Although many languages were promised when many languages were created, they were forgotten and shelved. However, even though C was created entirely from a personal need, it has achieved great success. Since the C language is written by an expert system programmer, it meets all the requirements of the system programmer. Other programming languages deviate from their original purpose when filling many unnecessary features for the convenience of the user, where the fine point is. While the language of personal need fulfilled its function, the languages created for trade disappeared.

2.2. What is C Programming Language and Recent History?

The first development stages of C took place at AT&T Bell Laboratories from 1969 to 1974. According to Ritchie, the most creative circuit was in 1972. Since many features of the language derive from a language called "B", the new language was called "C". The B language was interpreted and did not support the data type. Because the new hardware supports different data types, and the interpreted languages are relatively slow at runtime, the C language has been developed as B, with type support added and compiled.

Regarding the origin of the name "B", the rumors are different: Ken Thompson says B is derived from the BCPL programming language, but Thompson has also developed a programming language that he named Bon in honor of his wife Bonnie.

By 1973, C had become strong enough, and most of the UNIX kernel was rewritten with C, originally written in PDP-11/20 assembly language. Thus, UNIX was one of the first operating systems whose kernel was not written in an assembly language.

In 1978, Ritchie and Brian Kernighan published the first edition of "The C Programming Language". Known as "K&R" by C programmers, this book has been used as the unofficial standard of the C language over the years. This version of C is called "K&R C" today. The second edition of this book contains the ANSI C standard described below.

K&R C is generally considered the most basic part of the language that all C compilers must support. For many years, even after the adoption of ANSI C, when high portability is desired, K&R C has been considered "common denominator" by C programmers because some compilers have not yet been updated to support ANSI C A written K&R C program also supported ANSI C. Over the years following the release of K&R C, some "unofficial" features have been added to the language, supported by AT & T's compilers and some other computer manufacturers.

In the late 1970s, C began to outstrip BASIC as the most widely used microcomputer language. In the 1980s, its popularity began to grow with the adoption of the IBM PC. At the same time, Bjarne Stroustrup and his colleagues at Bell Laboratories began working to add orientation to the object. While C remains the most widely used language in the UNIX world today, Stroustrup developed and named C ++ the most important language in the Microsoft Windows operating system.

In 1983, the American National Standards Institute (ANSI) established a committee to establish a C standard. After a long and strenuous work, this board completed the standard in 1989 and the standard was published as ANSI X3.159-1989 "Programming Language C". This version of the language is often referred to as ANSI C. In 1990, this standard was adopted by the International Organization for Standardization (ISO) with minor amendments and published as ISO / IEC 9899: 1990.

One of the aims of the ANSI C was to establish a standard that included K&R C and added the "unofficial" features that were subsequently incorporated into the language. He added standard k function prototypes and a more capable preprocessor to the standard.

Today ANSI C is now supported by almost all compilers. The majority of C programs currently being written are written in accordance with the ANSI C standard. Only a program written using standard C can be compiled and run correctly with any standard compiler. However, programs written using non-standard libraries may require a specific platform or compiler.

C has, directly and indirectly, influenced and exemplified many languages. For example: C ++, Java, C #, Perl, PHP, JavaScript, Asp etc.

It is also shown in most programming languages, "Hello World!" The output is also the C programming language. This is the first example of output when learning programming languages. Thus, the C programming language has led and guided other languages with this example and learning method.

2.3. What is UNIX and Recent History?

UNIX-derived operating systems are flexible and robust tested under a wide range of conditions, from multi-processor to very expensive machines to single-processor simple and very cheap home computers. However, it has become a standard in multiprocessor servers especially with its stable structure and multi-user-multitasking structure and has found widespread use on workstations especially in the academic world. It spread rapidly between UNIX, Interdata 7/32, VAX, and Motorola 68000.

The Unix operating system was designed and implemented in 1969 at AT&T Bell Labs in the USA by Ken Thompson, Dennis Ritchie, Brian Kernighan, Douglas McIlroy, Michael Lesk, and Joe Ossanna. written in the translation language. It was rewritten in 1973 by Dennis Ritchie in the C programming language. The validity of an operating system written in a high-level language allows for easy portability to other different computer platforms. Due to a legal disruption that forced AT & T to license, UNIX grew rapidly and was recognized by teaching institutions and businesses.

UNIX is a multitasking computer operating system developed in Bell Labs in 1969 by Ken Thompson, Dennis Ritchie, Brian Kernighan, Douglas McIlroy, Michael Lesk, and Joe Ossanna. Communication between user and computer systems is provided using command interpreter software.

Linux, founded by Linus Torvalds, is not a UNIX but a UNIX derivative. It is an operating system kernel inspired by UNIX, developed by a group of independent software developers.

The MULTICS (Multiplexed Operating and Computing System) project developed by MIT, AT&T Bell Technology Laboratories and GE (General Electric) were established in 1960. The goal of the MULTICS project was to enable simultaneous data sharing by allowing multi-user access to the computer. and had developed a lot in time but had many problems.

In 1969, the project began to become complex and AT&T Bell Technology Laboratories withdrew. The reason for this is that MULTICS is now growing and decelerating beyond its objectives. However, in the same year, MULTICS began to make the job smaller and the first version of Unix was UNICS (Uniplexed Information and Computing System) by Ken Thompson, a researcher at Bell, who encoded a file system that simulated MULTICS software.). The first version was written in assembler, and it was a machine-only version.

In 1973, Thompson, together with Dennis Ritchie, the ancestor of the C compiler, created the 5th version of Unix by re-encoding the kernel with C. Thus, Unix has become a portable operating system consisting of codes that can be recompiled following various target hardware thanks to the portable C language.

2.4. What is Linux and Recent History?

Linux is one of the core components of computer operating systems. It is a free software project developed under the Linux Foundation and offered under the GNU General Public License. Linux was first named by Linus Torvalds in 1991. Today, it is widely used in the operating systems of supercomputers, smart devices, and devices used in internet infrastructure. The most popular of these is the Android operating system developed by Google.

The name Linux is also used as a common abbreviation to describe operating systems created using this kernel. For example, when the Linux kernel and the GNU tools are combined and presented as a complete operating system, it is called the GNU / Linux distribution but is referred to briefly as Linux in the spoken language.

In 1991, Linux was written from scratch by Linus Torvalds, a Finnish university student, following the architecture and POSIX standards of one of the older operating systems, UNIX. Although it was inspired by Unix architecture, there is no code from Unix in Linux. This software is a core software that does not have useful tools.

Long before Linux, Richard Stallman left his post at MIT in 1984 to develop a new UNIX-compatible operating system as a free software project. The operating system development project, called the GNU Draft, by 1991, the user tools were ready, but the core software was missing.

In 1992, Linus Torvalds decided to release the core software, which he had developed, under the GNU General Public License as free software to gain the support of more developers and contributors. Thus, these two projects (Linux kernel and GNU Draft) have completed each other's missing parts and can be presented as a complete operating system. This operating system was announced in 1994 in the GNU bulletin as "Free UNIX-Like".

Free software makes it legally possible for the software to be copied, changed and distributed following their licenses, and the software can be developed and shared by anyone who wants. Therefore, Linus' preference for the GNU General Public License is the most important breaking point in the history of Linux. In this way, the Linux project has been able to attract many volunteer experts from around the world. The free software movement, initiated by Richard Stallman, is also well known and recognized as a proven development model.

The Internet has opened a space for the needs of free software, providing the environment for developing software together universally. The GNU / Linux project has developed very well by taking advantage of experts from around the world from the 90s to the present.

Especially Apache software has made the internet servers faster and more stable and cost-effective than their competitors. This has contributed greatly to the commercial and technological development of Linux systems since the late 90s.

Besides, the widespread use of the internet in the 2000s to homes and small businesses has increased the need for very broad internet infrastructure and very different network devices.

The fact that the GNU / Linux system is infinitely customizable by device manufacturers and offered free of charge has made them widely preferred. Running the Linux kernel alone does not provide a graphical desktop environment. Many software needs to be brought together.

In the coming period, the fact that commercial Linux distributions were generating revenue from servers and mobile systems rather than desktops led to limited support for developers and companies for the desktop environment. However, nowadays Linux systems have advanced desktop technologies such as GNOME, KDE, Xfce.

Systems using Linux kernel have a market share of approximately 2% in the desktop, notebook and netbook market. It is mostly preferred by software developers, computer experts, and free software volunteers.

Examples of desktop Linux systems include Ubuntu, Debian, Fedora. Linux distributions are being developed to address the end-user. The user interface, GNOME, KDE, a desktop environment such as Xfce, Mozilla Firefox, Chromium a web browser, LibreOffice office software set such as video-music player, CD / DVD burner, graphics processing software and so on. popular free software of this kind is packaged and presented to the end-user.

The world's top 10 supercomputers use Linux. As of November 2010, 459 (91.8%) of the top 500 systems are using Linux. It was also selected as the operating system for IBM Sequoia, the world's most powerful supercomputer, which was launched in 2011.

Most of the companies that produce commercial PC games have not produced games for Linux systems until recent years. For this reason, the gaming possibilities on Linux systems are very limited. The DirectX library used by the game software does not work on Linux systems. The OpenGL library running on Linux is not preferred by the manufacturers.

However, with the launch of the Linux client on the Steam gaming platform in 2012, Linux versions of some popular games have begun to be produced. There is also a wide range of games available in the Android application store.

Major car manufacturers such as Toyota, Nissan, Jaguar, Land Rover, Ford, Mazda, Mitsubishi, and Subaru have been using Linux for a long time in the digital systems of their vehicles. The Automotive Grade Linux project is being implemented by the Linux Foundation to produce smart cars, with major technology and automotive manufacturers becoming members of the project.

2.5. What is Shell Script and Recent History?

Bash (Shell) Script is one of the most popular programming tools available on Unix. Stands for Bourne Again Shell. It is a powerful tool for all Linux users or System Administrators.

Bash is an extremely powerful and useful piece to develop scripts. Repeated tasks can fit into a single-line function call. Thus, many long commands can be reduced to a single executable command.

The Bash script is available on almost all Linux distributions and does not require a separate installation.

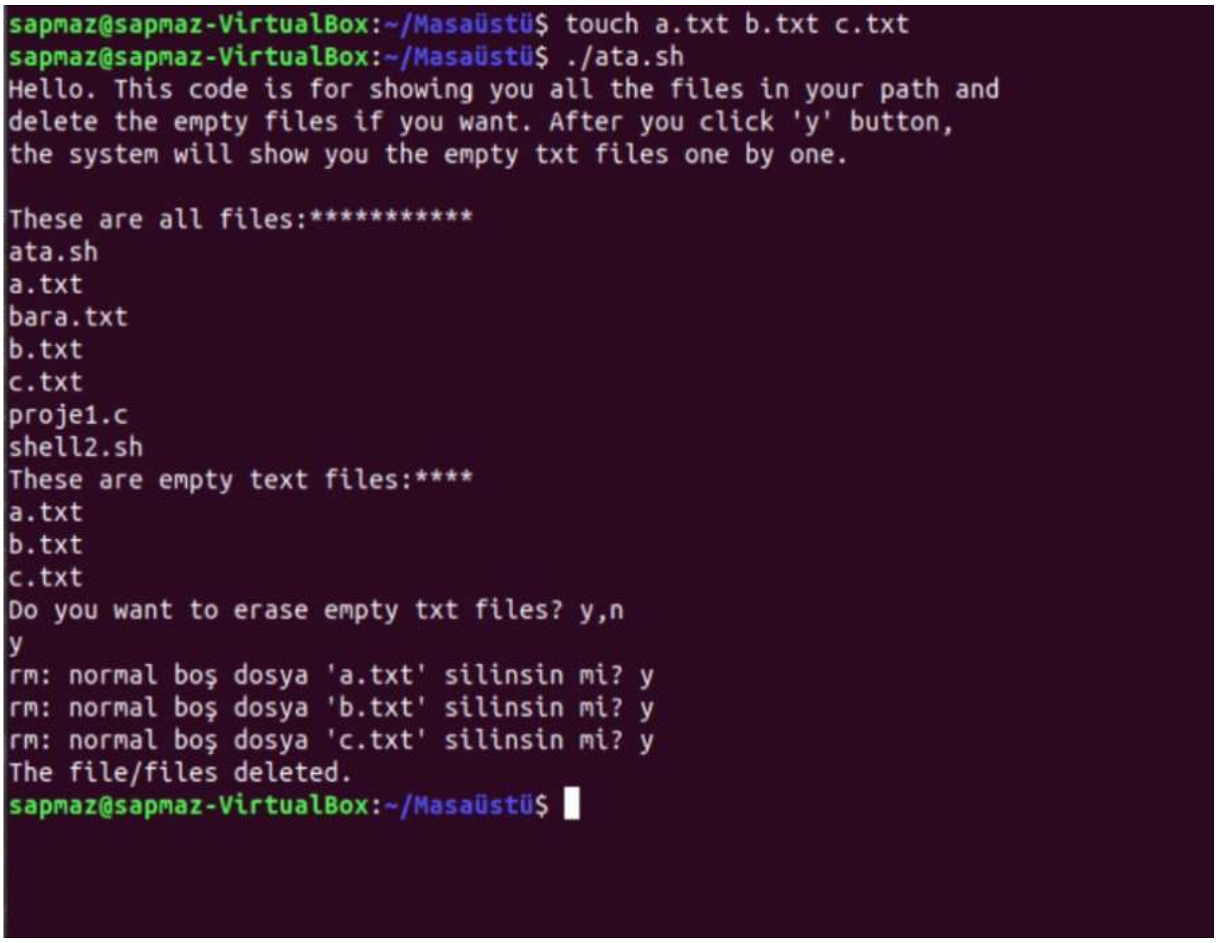
You can eliminate repetitive tasks, save time, achieve a well-structured, modular and formatted sequence of events, provide dynamic values to commands using command-line arguments with functions, reduce complex commands to a single command that runs, run everyone as many times as needed, create logical streams using bash scripts can be installed, bash functions can be run at server startup or by adding a scheduled task, debugging commands, having interactive shell commands, bash script is a great tool to simplify your workflow and improve your project, and their potential for use is unlimited.

3. Our Project on Shell Script

3.1. Deleting Empty .txt Files

This is our first shell code. Wherever we run the .sh file, it displays all the files there first. Then we kept the empty text files in an array. We decided to write such code because it would be very difficult to get into each empty file and look one by one. This code can of course be improved by making it available for directries or any file that is empty. Then we printed the array that holds the empty text files, and the user will see the empty files here and specify which one to delete by typing yes or no. And he can delete the files he wants.

Here is an output example for the “Deleting Empty .txt Files” project:



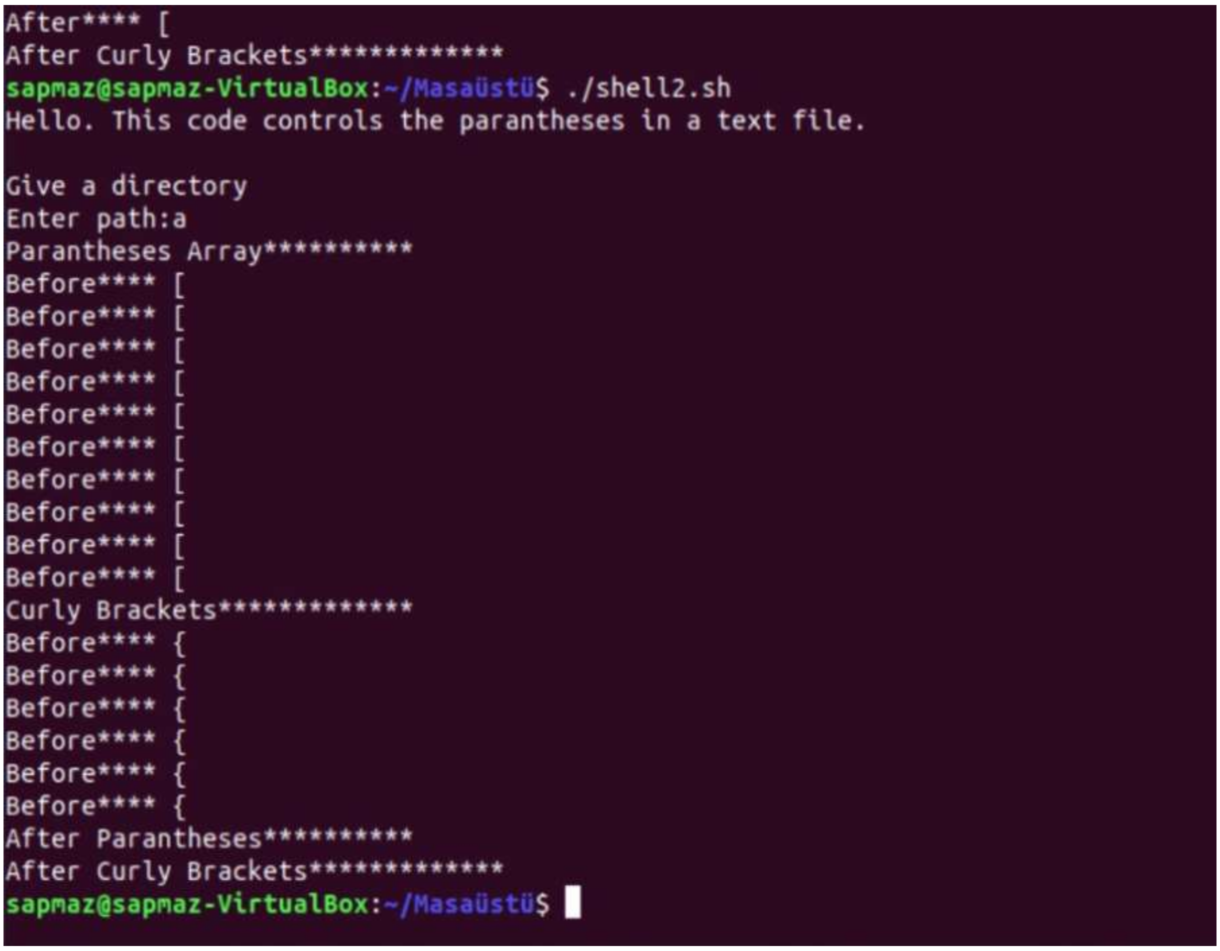
```
sapmaz@sapmaz-VirtualBox:~/Masaüstü$ touch a.txt b.txt c.txt
sapmaz@sapmaz-VirtualBox:~/Masaüstü$ ./ata.sh
Hello. This code is for showing you all the files in your path and
delete the empty files if you want. After you click 'y' button,
the system will show you the empty txt files one by one.

These are all files:***********
ata.sh
a.txt
bara.txt
b.txt
c.txt
proje1.c
shell2.sh
These are empty text files:****
a.txt
b.txt
c.txt
Do you want to erase empty txt files? y,n
y
rm: normal boş dosya 'a.txt' silinsin mi? y
rm: normal boş dosya 'b.txt' silinsin mi? y
rm: normal boş dosya 'c.txt' silinsin mi? y
The file/files deleted.
sapmaz@sapmaz-VirtualBox:~/Masaüstü$ 
```

3.2. Parentheses Checker

This is our second shell code. It's looking for brackets. It checks to see if each open bracket has closed a closed bracket. If it is closed, there is no parenthesis left after situation. If not, the parentheses appear after situation. We did this code with arrays. The lack of a stack in Shell made us very difficult. If there was a stack, we would do it with stack commands, but we didn't have to do it with arrays. This code can actually be developed for necessary functions such as if-fi do-done.

Here is an output example for the “Parentheses Checker” project:



```
After**** [
After Curly Brackets*************
sapmaz@sapmaz-VirtualBox:~/Masaüstü$ ./shell2.sh
Hello. This code controls the parantheses in a text file.

Give a directory
Enter path:a
Parantheses Array**********
Before**** [
Before**** [
Before**** [
Before**** [
Before**** [
Before**** [
Before**** [
Before**** [
Before**** [
Before**** [
Curly Brackets*************
Before**** {
Before**** {
Before**** {
Before**** {
Before**** {
Before**** {
After Parantheses**********
After Curly Brackets*************
sapmaz@sapmaz-VirtualBox:~/Masaüstü$ 
```

4. Our Project on C Programming

4.1. Password for Files

Our third code is a C code. Lock a file. Our goal is for the user to lock a file and set a password to open it. You have the right to enter this password three times. At first we will show the file to half. The password will be required to view the rest later. And if the user enters the wrong password 3 times, that file will self-destruct. We used file opearations when we tried to write this code.

4.2. Header Files Creator

Our fourth and final code is to make header files. If we had time, that's the point of our fourth code. Header files are normally very simple, but they are a bit of a hassle. We wanted to analyze a c code, look at the prototypes (or understand the code if there are no prototypes) and write a c code that creates header files according to the functions used in the code.

5. Discussion

We tried to write the 4-command idea we found for the parts that can be forced by a shell and c author or for the parts that can make it easier. Shell and c are important to program the system. And we've tried to make the little touches that need to be done in order to reduce the difficulty of system programming or reduce time loss.

6. Conclusion

In conclusion, shell and c code are important codes for programming a system. Since we can access the hardware with shell scripts, this gives us a great deal of convenience. When writing a program or accessing the system, it is very necessary to know Shell well to communicate with the Kernel.

7. References

[1] Ritchie D. M. & Johnson S. C. & Lesk M. E. & Kernighan B. W. “UNIX time-sharing system: The C Programming Language”, Nokia Bell Labs, U.S, 1978.

[2] Andersen L. “Program Analysis and Specialization for the C Programming Language”, University of Copenhagen, Denmark, 1994.

[3] Kelley A. & Pohl I. “A book on C; C Programming in C”, Benjamin-Cummings Publishing Co., California, 1994.

[4] Devietti J. & Blundell C. & Martin M. & Zdancewic S. “Hardbound: architectural support for spatial safety of the C programming language”, Proceeding, Seattle, 2008.

[5] Stevens W. & Rago S. “Advanced Programming in the UNIX Environment”, Addison Valley, U.S, 2001.s

8. Who did What

We did together all of them, but as follows:

- Ata: Shell1, Report
- Dilay: Shell1, Shell2, PowerPoint, C1, Comments
- Nur: Shell1, Shell2, C1