**Subject with samples**

There are no restrictions on the project’s subject. However, to alleviate any confusion that may arise during the subject selection process, a few sample project subjects will be introduced briefly.

NOTICE: Do not copy the sample project ideas ‘as they are’. It will result in a deduction of points.

1. ls -R1fF

In this subject, you need to create a program that behaves exactly the same as executing the ‘ls’ command with the options -R1fF. The syntax of execution is following:

$ ./lsr1ff [file\_or\_dir ..]

for example, the command

$ ./lsr1ff /etc /bin/who /abc/def

lists recursively all files and directories under the /etc directory, displays the name /bin/who with an asterisk appended, and reports that /abc/def does not exists (in case that there isn’t /abc/def).

1. myfind

In this subject, you will write a program that works a little bit like the Unix **find** command. In doing so, you will have a chance to work with the UNIX directory structure, recursion, and optionally, the stat() system call.

The UNIX **find** command searches directories and all their subdirectories for files that fit certain specifications. **find** also allows you to search for files by modification time, by size, by owner, by number of links, as well as combinations of attributes.

The original find command example is following:

$ find /home -name Makefile -print

As mentioned before, **find** command has more options to use and complicated syntax. However, student version can be much simpler like taking only two arguments only.

To implement this project, think what ‘myfind’ has to do. If the entry matched the name you are looking for, print the directory name with the file name appended. If the entry is a directory your program will have to do the same operations on that subdirectory. There for the search and print function will be recursive.

Consider many things to make your ‘myfind’ work as same as the shell command **find** does.

1. pong

In this subject, you will write a video game based on the classic coin-operated computer game ‘pong’. This example is a one-person ping-pong game; the object of the game is to keep the ball in plays as long as possible. In implementing this game, you will work with screen management, signal and timer management, and the random number function.

스크린샷, 라인, 직사각형이(가) 표시된 사진

자동 생성된 설명

(Figure 1) ‘pong’ playing screen example.

The ball is served from the middle of the court with a random speed. The ball bounces elastically against the walls. The paddle, though, introduces some uncertainty to the game. Each time the ball bounces off the paddle, the speed of the ball is changed by a small, random, amount.

The player moves the paddle up and down on the screen, one row at a time, by pressing arrows or keys you set.

The paddle won’t move beyond its top and bottom limits.

At any time pressing key ‘Q’, the play would quit from the game.

1. sumac

Email is one of the two main reasons people use the Internet. When someone sends you a piece of email, that letter is stored in a mailbox on a computer somewhere. When you check your email, your mail-reading program tells you what is in your mailbox and allows you to retrieve letters from your mailbox.

In this subject, you will write a mail access program. You can use it to list, read, and save letters.

In the process, you will learn how client-server systems work and you will be able to reuse many of ideas and skill you have learned earlier in this course.

You may start from scratch and design this any way you please. To implement the project, you need at least two programs respectively – client and server.

From a mail system perspective, the server is responsible for storing or deleting emails for multiple users and facilitating the sending of emails to other users. On the other hand, the client can download incoming emails for themselves from the server or compose emails to send to the server, which will then be delivered to other users.

In addition, user interface is up to you. Let me introduce one possible presentation:

텍스트, 폰트, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명

(Figure 2) mail system for client example

Consider ‘curses’ or ‘tcsetattr’ or else to implement like above.

1. mywatch

For this subject, you will write a program that implements some of the functions of the **watch** feature of the UNIX shell.

Sometimes you are waiting for one or more users to login to the system. There are lots of reasons why you might want to know when someone logs in. You might want to start a chat session with that user. You might be waiting for that person to login and download a file you need. The **who** command might be useful; it tells you which users are logged in right now. You could type ‘who’ every few minutes and see if the logname appeared on the list. That would be pretty tedious.

For example, write a program called ‘mywatch’ that takes as command line arguments a list of users you want the program to watch for. Every five minutes the program wakes up and checks the utmp file. It compares the list of active users it finds there to the list of active users it found last name. If it finds a user is not logged on but had been logged on before, it should tell you. If it finds a user is logged on now and had not been logged on before, it should tell you.

The program should produce output of the form like (the … indicates five-minutes pauses):

텍스트, 영수증, 대수학이(가) 표시된 사진

자동 생성된 설명

(Figure 3) ‘mywatch’ to monitor user login example