HACETTEPE UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING

BBM-203 PROGRAMMING LAB.
ASSIGNMENT 4

Subject : UNIX C Programming
Submission Date : 10.12.2012
Deadline : 28.12.2012
Programming Language : C
Operation System : CentOS 5.8 (gcc 4.1.2)
Advisors : Dr. Burcu Can, Dr. Sevil Şen, R. A. Çağdaş Baş

1. Background Information:

UNIX has become one of the most popular operating systems since it was first developed as a multiuser, multitasking operating system in the early 1970s. Today, there are many versions of UNIX. Since there is no operating system called "UNIX", instead of saying "UNIX is an operating system", it's more appropriate to say "there are operating systems, which are versions (or derivatives) of UNIX". Some popular UNIX versions are System V (from AT&T), BSD (Berkeley Software Distribution) UNIX and Linux. Linux is a free operating system for anyone to use, modify and distribute. It is developed by a large community of people who work on open source software and standards. By now, you must have been familiar with at least one Linux distribution (Fedora, for example).

As an operating system user and surely, as a system programmer, you'll use different capabilities of UNIX in two different fields:

- Session environment
- Programming environment

The session environment is managed by shells. A shell is a command interpreter that acts as an interface between users and operating system kernel (kernel is the layer that takes part over the machine hardware and responsible for main operating system functions such as memory management and file system management). There are many shells in UNIX, including sh (Bourne shell), csh (C shell) and bash (Bourne-again shell). When a user enters a command at a terminal, the shell interprets the command and calls the related program. In addition to its function of interpreting commands from a terminal and sending them to the operating system, the shell can be used as a high-level programming language. Shell commands can be arranged (for example in a file) for later execution as a high-level program. This process is known as shell programming.

The UNIX programming environment mainly consists of a rich set of programming languages (including C, C++, Ada, Java, etc.) and development tools. The C programming language is the most popular programming language to use with UNIX, because the operating system itself is written mostly in C.

2. PROBLEM:

The primary goal of this experiment is to make you acquainted with UNIX programming environment. You will design and implement an application program in ANSI C and compile it with *gcc*. You will also use the *make* utility and write a *makefile* [4] for your project.

The application is a file listing program (*list*). The program will list directories, subdirectories and files of a path recursively. *list* will get the path or path pattern as argument. Output list has to be in alphabetical order.

The formal definition of *list* is given below:

NAME

list – list directory contents in alphabetical order.

SYNOPSIS

list [options] [path OR pattern]

DESCRIPTION

list lists given path's contents. *list* shall write the names of files contained within a directory as well as any requested, associated information.

If no operands are specified, *list* shall write the contents of the current directory.

OPTIONS

The following options shall be supported:

--help

Output this help message

-a, --all

Write out all directory entries, including those whose names begin with a period ('.'). Entries beginning with a period shall not be written out unless explicitly referenced.

-d, --directory

List only directories

-f, --file

List only files.

-h, --human-readable

With -l, print sizes in human readable format (e.g., 1K 234M 2G)

-l

(The letter ell.) Write out in long format, displaying UNIX file types, permissions, number of hard links, owner, group, size, date, and filename

-r, -R, --recursive

Recursively list subdirectories encountered.

USAGE

Suppose that you have a file hierarchy as given below.

```
| -- <temp>
|      | -- <BBM>
|      |      | -- exp 3.txt
|      |      | -- last code.c
|      |      | -- exp 4.txt
|      | -- <BIL>
|      |      | -- odev.dat
|      |      | -- experiment.sh
| -- <windows>
|      | -- .forward
```

Here are some usage examples:

```
$ list –a temp
.
. .
BBM
BIL
$ list –l
drwx------ 1 Administrators None   0 Sep 22 20:07 temp
drwxr-xr-x 1 cagdas         None   0 Jul 01 10:27 windows
$ list –ahl temp/BBM
drwx------ 1 Administrators None  4,0K Aug 29 01:38 .
drwx------ 1 Administrators None  4,0K Aug 29 01:38 ..
-rwx------ 1 Administrators None   446 Oct 22  2009 exp 3.txt
-rwx------ 1 Administrators None  1,7K Dec 22  2009 last code.c
-rwx------ 1 Administrators None  9,2M Nov 22  2009 exp 4.txt
```

```
$list -r
./temp :
BBM
BIL
./temp/BBM :
exp3.txt
last code.c
exp 4.txt
./temp/BIL :
odev.dat
experiment.sh
./windows :
$ list temp/B*
./temp/BBM :
exp3.txt
last code.c
exp 4.txt
./temp/BIL :
odev.dat
experiment.sh
$ list te?p
BBM
BIL
$ list -a windows
.
..
.forward
```

Please inspect UNIX standard function *ls* for detailed usage. All parameters except –f and –r (only lowercase r) are the same with *ls.*

PATTERNS

As mentioned above, your program will accept a pattern instead of a direct path as an argument. This means given abstract string can match to a number of strings.

You will handle two types of patterns in this experiment:

- Star (“*”) : any number of alpha-numeric character (0..∞)
- Question Mark (“?”) : One and only one alpha-numeric character

These marks can be used in various ways to express a pattern. These marks can occur at the beginning, middle or the end of a sentence. Here are some examples for “star”:

“B*” will match with: “B”, “B1” “BILGISAYAR-2” and so on, and will not match with “bilg”, “ABC”

“B*L” will match with: “BIL”, “BASDFG-23L” and so on, and will not match with “BBM”, “B”

“*L” will match with “BIL”, “BBML”, “L” and son on, and will not match with “BBM”, “B”

Here are some examples for “Question Mark”:

“B?L” will match with “BIL”, “B1L” and so on, and will not match with “BL”, “BBML”

“?IL” will match with “BIL”, “1IL” and so on, and will not match with “IL”, “BBIL”

“BI?” will match with “BIL”, “BI1” and so on, and will not match with “BI”, “BILL”

WORKING WITH LINUX

There are various ways to work on different platforms. You can always send your project files with a FTP program like WinSCP [5], and compile and run on console with a SSH program like Putty [6].

Also you can use remote development tools which come built-in with popular IDEs like Netbeans [7] [8].

Another option is to use Cygwin tool for windows [9]. It is a collection of tools which provide a Linux look and feel environment for Windows. Please be aware that *ls* outputs on Cygwin can differ from dev machine.

Please make sure that your program is running smoothly on dev machine or you will absolutely not be graded. Outputs have to be exactly the same with *ls* outputs on dev machine.

HINT

You will implement your program in C. You have to perform file and directory operations using UNIX system calls. You may want to use the following system calls functions:

open(), close(), lseek(), read(), fcntl(),write()
stat(), fstat(), lstat(), unlink()utime(), chdir(),
getcwd(), opendir(), readdir(), closedir(),rmdir()

IMPORTANT ISSUES

- Test your program on "dev.cs.hacettepe.edu.tr" before submission.
- **Projects without a proper Makefile won't be graded.**
- Your output has to be in alphabetical order.
- **DO NOT** use ***"listdir"*** or ***"system"*** functions.
- Do not use *fopen()* or *fclose()* functions; use *open()* and *close()* functions instead if needed.
- Use *dirent.h, stat.h* libraries and related libraries for file and directory operations.

SUBMISSIONS:

Your submission will be in the format below:

<studentid>.zip

```
-- report
        | - report.pdf
-- source
        | - *.c
        | -*.h
        | Makefile
```

LAST REMARKS:

- Regardless of the length, use **UNDERSTANDABLE** names to your variables, classes and functions.
- Write **READABLE SOURCE CODE** block
- Your submission code file structure is going to be announced later.
- **You will use online submission system to submit your experiments. https://submit.cs.hacettepe.edu.tr/ Deadline is: 23:59 pm. No other submission method (such as diskette, CD or email) will be accepted.**
- Do not submit any file via e-mail related with this assignment.
- **SAVE** all your work until the assignment is graded.
- The assignment must be original, **INDIVIDUAL** work. Duplicate or very similar assignments are both going to be punished. General discussion of the problem is allowed, but **DO NOT SHARE** answers, algorithms or source codes.
- You can ask your questions through course's news group and you are supposed to be aware of everything discussed in the newsgroup: **news://news.cs.hacettepe.edu.tr/dersler.bbm203**

REFERENCES

[1] T. Cormen, C. Leiserson, R. Rivest, C. Stein, Introduction to Algorithms, Second Edition, The MIT Press, 2003 (Clrs)
[2] www.cplusplus.com
[3] http://linux.about.com/od/commands/l/blcmdl.htm
[4] http://web.mit.edu/gnu/doc/html/make_3.html
[5] http://winscp.net
[6] http://www.putty.org/
[7] http://netbeans.org/
[8] http://netbeans.org/kb/docs/cnd/remotedev-tutorial.html
[9] http://www.cygwin.com/