Wollongong of University
School of CS & SE, Faculty of Informatics

CSCI212/MCS9212 Interacting Systems Autumn 2013

Assignment 1 (7.5 marks)

Due 11:59pm Friday March 29, 2013.

Aim

The aim of this assignment is to become familiar with the Unix file system, Unix pipelines and the implementation of filters. This assignment should also help you to remember what you did in first year.

Introduction

This assignment concerns the writing of a program which provides statistical information of the directory contents based on the `ls` command. The `ls` command is a command for listing files in Unix and Unix-like operating systems (e.g. Linux, Sun). `ls` is an abbreviation of *list segments* and is specified by POSIX and the Single UNIX Specification. Your job is to write a filter (C/C++ program) to take input from the `ls` command and process it to yield a report.

For the purposes of this assignment you are to use the `ls` command with the `-l` option.

.

```
   -l      Lists in long format, giving mode, ACL indication, number of links, owner,
           group, size in bytes, and time of last modification for each file. If the
           file is a special file, the size field instead contains the major and minor
           device numbers. If the time of last modification is greater than six months
           ago, it is shown in the format `month date year' for the POSIX locale.
           Files modified within six months show `month date time'. If the file is a
           symbolic link, the filename is printed followed by "->" and the path name
           of the referenced file.
```

Permissions	Number of links	Owner	Group	Size	Time	Directory or file
-rwxr--r--	1	User1	root	26	Mar 2 19:51	MyFile.txt
drwxrwxr-x	3	User2	csstf	4096	Oct 15 2008	MyDir

To increase the scope and power of the ls command you may also use the `-R` and `-a` options:

```
   -R      Recursively traverses subdirectories encountered.

   -a      Lists all entries, including those that begin with a dot (.) which are
           normally not listed.
```

Please note: using of the -R option can cause the `ls` command to traverse a large portion of the file system which can take considerable time and CPU expense. To halt the `ls` command just press the *'control'* and *'c'* keys simultaneously `<crl><c>`.

Below is a brief description of each of the above categories shown when using the `ls -al` command. (Refer to man page of `ls` command for more details.)

Permissions - The permissions of the directory or file are explained below.

The first character in the first column is '`d`', which means it is a directory. The other entries here are files, as indicated by the '-'.

```
 d            rwx         r-x          ---
file type    users       group        others
```

The next 9 characters define the file permissions. These permissions are given in groups of 3 each. The first 3 characters are the permissions for the owner of the file or directory. The next 3 are permissions for the group that the file is owned by and the final 3 characters define the access permissions for everyone not part of the group. There are 3 possible attributes that make up file access permissions.

r - Read permission. Whether the file may be read. In the case of a directory, this would mean the ability to list the contents of the directory.

w - Write permission. Whether the file may be written to or modified. For a directory, this defines whether you can make any changes to the contents of the directory. If write permission is not set then you will not be able to delete, rename or create a file.

x - Execute permission. Whether the file may be executed. In the case of a directory, this attribute decides whether you have permission to enter, run a search through that directory or execute some program from that directory.

Number of links - The amount of links or directories within the directory. The default amount of directories within a directory is always 2 because of the . and .. directories.

Owner - The owner of the file or directory

Group - The group assigned to the file or directory

Size - Size of the file or directory.

Time - Time of last modification.

Directory or file - The name of the directory or file.

Implementation

Using the output of the command `ls`, your program should read this output as standard input (e.g. cin). In other words your program is to behave as if it were part of a pipeline. Typical execution could be as follows:

`ls -l | ./a.out` OR **`ls -alR | ./a.out`**

where `a.out` is your executable program which will take in the `stdout` of `ls` command as `stdin` of your program. To do this treat the ouput from `ls -l` as columns and process the output line by line.

Your program should read the output from `ls -l` and do the following.

1. For each user in the entire output process table, print the user name of the user (in alphabetic order), followed by: the number of files owned by the user, the oldest file, the latest file and the total size of all the user's files. Note: you should ignore any lines describing directories, links, doors or any lines without file permissions (eg error messages). The total size of the files should be expressed as bytes, KB, MB, GB or TB accurate to 4 significant places (e.g. 4.321KB). Below shows an example output:

```
User        Files        Oldest         Latest       TotSize

alpha1        511    Apr 23  2005    Feb 28 01:50    1.013GB
root          102    Jul  3  1970    Sep 21  2012    5.732MB
xyz01          10    Jun 30  2005    Jan  2 19:51    1.347KB
```

2. After printing the user file details (above) leave one blank line and print the stats for all users as shown below:

```
ALL           623    Jul  3  1970    Jan  2 19:51    1.018GB
```

Data Structures

In order to do this assignment, you will need to implement appropriate data structures to store the data. The main data storage structure will be a linked list of nodes containing user info. To maintain alphabetic order you will need a find() and insert() function based on the user name. Once all the data is inserted into the linked list your program should iterate the linked list and produce the output, as described above.

Process

1. Design a structure for a list node, which will typically store appropriate information about users. (1 mark)

2. Implement the linked list. The linked list should use the node you designed in step one. Put the implementation of your linked list in linkedlist.cpp and the header in linkedlist.h. (Preferably use classes to do this. Do not use the standard template library for this.) (3 marks)

3. Implement main.cpp, which will read ls -l from standard input and perform the appropriate manipulation. Once all input is processed, produce the output as mentioned above. (3.5 marks).

Memory should be cleaned up prior to termination.

The compile directive will be:

```
CC linkedlist.cpp main.cpp -o ass1
```

Tips

Before you commence this assignment, you should have a look at how the ls program behaves. Run it several times with different options in different directories on different hosts (i.e. `banshee` and `linux`). This will help you process the input streams more effectively.

Submit:

Submit the source files `linkedlist.cpp`, `linkedlist.h` and `main.cpp` using the submit program. The directive is as follows:

```
submit -u YourUserName -c csci212 -a a1 linkedlist.cpp linkedlist.h main.cpp
```

An extension of time for the completion of the assignment may be granted in certain circumstances. A request for an extension must be made to the Lecturer before the due date. Late assignments without granted extension will be marked but the mark awarded will be reduced by 10% for each day late. Assignments will not be accepted more than three days late.