

CSCI 352 Unix Software Development Summer 2015 Assignment 3

Submitting Your Work

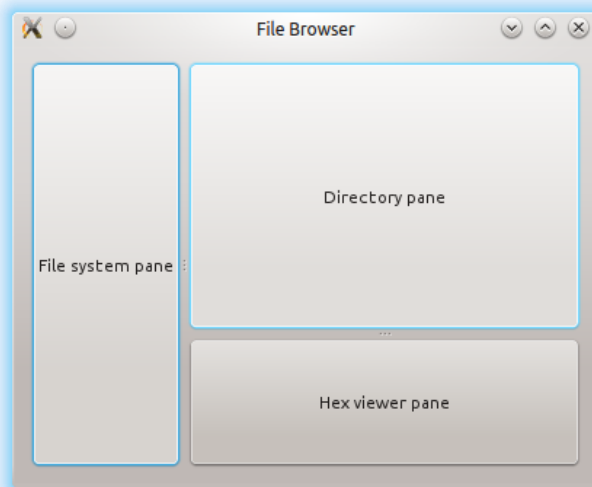
This assignment is worth 15% of the grade for the course. Save your program files (including header files and makefile) in the zipped tar file WnnnnnnnnAssg3.tar.gz (where Wnnnnnnnn is your WWU W-number) and submit the file via the **Assignment 3 Submission** item on the course web site. You must submit your assignment by 9:00am on Monday, August 17.

For this assignment, you are to implement a file browser, providing a graphical user interface through GTK+3.0.

Your assignments will be evaluated on correct functionality and conformance to the coding standards described at the end of this assignment specification.

The File Browser

The file browser must have three resizable panes, organized as shown below.



- The left pane (the “file system view”) must provide a hierarchical view of the file system of the computer on which it is running.
- The top-right pane (the “directory view”) must list the files in whichever directory is selected in the file system view.

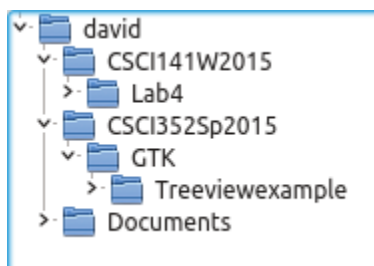
- The bottom-right pane must contain a hexadecimal view of the data in whatever file is selected in the directory view.

Each of the panes must provide vertical and horizontal scrollbars whenever needed to display their contents.

The File System View

The File System view must show the hierarchy of directories in the file system, starting with the root directory or the user's home directory.

Except when displaying the root directory, the file system view must contain an entry for “..”, the parent directory of the currently displayed directory.

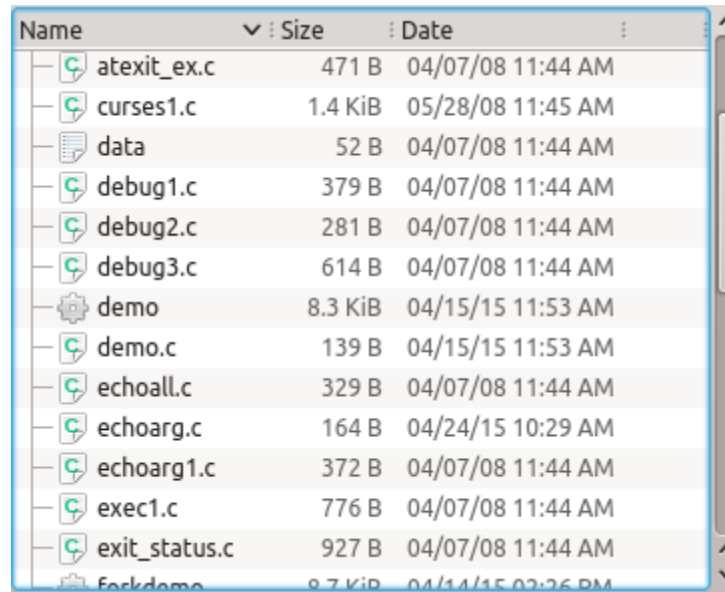


Selection of a directory in this view must result in display, in the directory view, of the list of files and subdirectories of the selected directory.

The Directory View

The Directory view must list the files and subdirectories in the directory selected in the File System view. For each file display information in sortable columns:

1. An icon representing the file type with its name.
2. The size in bytes.
3. The last modification date and time.
4. (This column should not be sortable) The access permissions for the owner, group and other users, displayed in rwx form, as in the Unix `ls -l` command.



Name	Size	Date
atexit_ex.c	471 B	04/07/08 11:44 AM
curses1.c	1.4 KiB	05/28/08 11:45 AM
data	52 B	04/07/08 11:44 AM
debug1.c	379 B	04/07/08 11:44 AM
debug2.c	281 B	04/07/08 11:44 AM
debug3.c	614 B	04/07/08 11:44 AM
demo	8.3 KiB	04/15/15 11:53 AM
demo.c	139 B	04/15/15 11:53 AM
echoall.c	329 B	04/07/08 11:44 AM
echoarg.c	164 B	04/24/15 10:29 AM
echoarg1.c	372 B	04/07/08 11:44 AM
exec1.c	776 B	04/07/08 11:44 AM
exit_status.c	927 B	04/07/08 11:44 AM
forkdemo	9.7 KiB	04/14/15 02:26 PM

The Directory view must be sortable on any of the first three columns. Selection of a file from the directory view must result in display of the contents of that file in the hexadecimal viewer.

The Hexadecimal Viewer

The hexadecimal viewer shall display the raw byte values from the file in hexadecimal notation, in rows of 16 bytes per row. The display is to be in three parts:

1. The leftmost part shows the start *address* of the row (its offset, in bytes, from the start of the file), in hexadecimal
2. The central part shows the value of each byte in two hexadecimal digits. Pairs of hexadecimal digits are to be separated by a single space.
3. The rightmost part shall show the character representation of the bytes in the row; for those bytes that do not have a printable representation, display a dot.

```

00000000 23 69 6E 63 6C 75 64 65 20 3C 73 79 73 2F 77 61 #include <sys/wa
00000010 69 74 2E 68 3E 0A 23 69 6E 63 6C 75 64 65 20 3C it.h>.#include <
00000020 73 74 64 6C 69 62 2E 68 3E 20 0A 23 69 6E 63 6C stdlib.h> .#incl
00000030 75 64 65 20 3C 73 74 72 69 6E 67 2E 68 3E 20 0A ude <string.h> .
00000040 23 69 6E 63 6C 75 64 65 20 3C 73 74 64 69 6F 2E #include <stdio.
00000050 68 3E 20 20 20 0A 23 69 6E 63 6C 75 64 65 20 3C h> .#include <
00000060 75 6E 69 73 74 64 2E 68 3E 0A 23 69 6E 63 6C 75 unistd.h>.#inclu
00000070 64 65 20 3C 73 79 73 2F 74 79 70 65 73 2E 68 3E de <sys/types.h>
00000080 0A 23 69 6E 63 6C 75 64 65 20 3C 65 72 72 2E 68 .#include <err.h
00000090 3E 0A 23 69 6E 63 6C 75 64 65 20 3C 73 74 64 61 >.#include <stda
000000A0 72 67 2E 68 3E 0A 0A 69 6E 74 0A 6D 61 69 6E 28 rg.h>..int main(
000000B0 69 6E 74 20 61 72 67 63 2C 20 63 68 61 72 20 2A int argc, char *
000000C0 61 72 67 76 5B 5D 29 0A 7B 0A 09 69 6E 74 20 66 argv[]){..int f
000000D0 64 5B 32 5D 3B 0A 09 70 69 64 5F 74 20 70 69 64 d[2];..pid_t pid
000000E0 3B 0A 09 0A 09 63 68 61 72 20 2A 61 72 67 5B 32 ;...char *arg[2
000000F0 5D 3B 0A 09 63 68 61 72 20 2A 73 72 67 5B 33 5D ];..char *srg[3]
00000100 3B 0A 09 61 72 67 5B 30 5D 20 3D 20 28 63 68 61 ;..arg[0] = (cha
00000110 72 20 2A 29 63 61 6C 6C 6F 63 28 33 2C 20 73 69 r *)calloc(3, si
00000120 7A 65 6F 66 28 63 68 61 72 29 29 3B 0A 09 73 74 zeof(char)).. st

```

Saving your files in a zipped tar file

You need to submit all your C and header files and a makefile that can be used to compile and link your programs. First you need to bundle them up into a single *tar* file. The term “tar” is an abbreviation of “tape archive” and goes back to the days when people would save a back-up copy of their files on magnetic tape. Nowadays, with the price of large disk drives so low, nobody uses tape anymore, but the concept of tar files survives.

1. Use the command:

```
tar -cf WnnnnnnnnAssg1.tar *.c *.h Makefile
```

(where Wnnnnnnnn is your W-number).

The -cf specifies two options for the tar command: 'c' means create and 'f' means that the name of the resulting tar file comes next in the command.

Following the name of the tar file, we list the files to be included in the tar file. In this case, we want all the files in your current directory whose names end with “.c” or “.h”.

2. If you now use the command `ls` you should now see the file WnnnnnnnnAssg1.tar in your directory.
3. If you use the command

```
tar -tf WnnnnnnnnAssg1.tar
```

(where Wnnnnnnnn is your W-number), it will list the files within the tar file.

4. Now compress the tar file using the gzip program:

```
gzip WnnnnnnnnAssg1.tar
```

5. By using the `ls` command again, you should see the file WnnnnnnnnAssg1.tar.gz in your directory. This is the file that you need to submit through the **Assignment 1 Submission** link in the course web site.

Coding Standards

1. Use meaningful names that give the reader a clue as to the purpose of the thing being named.

2. Avoid the repeated use of numeric constants. For any numeric constants used in your program, use `#define` preprocessor directives to define a macro name and then use that name wherever the value is needed.
3. Use comments at the start of the program to identify the purpose of the program, the author and the date written.
4. Use comments at the start of each function to describe the purpose of the function, the purpose of each parameter to the function, and the return value from the function.
5. Use comments at the start of each section of the program to explain what that part of the program does.
6. Use consistent indentation.