Systems Development Assignment 2, Diet 1 Coursework 2 18/19

Callum Carmicheal, S1829709

Table of Contents

1. Documentation

1.1. Notes and Files

List of files:

File name	***Description***	***Command*** (any relevant commands)
`client.c`	The client source code with all comments stripped from the file. (***Comments removed***).	`make`
`client-comments.c`	The client source code with all original comments left in.	`make -f Makefile-build-comments`
`documentation.docx`	The documentation and notes file that describes the compilation instructions, user guide and file notes.	-
`Makefile`	The Makefile that compiles uncommented source files.	`make`
`Makefile-build-comments`	The Makefile that compiles the source files with comments.	`make -f Makefile-build-comments`
`rdwrn.c`	The network source code for read and writing.	-
`rdwrn.h`	The header file that defines the networking functions.	-
`server.c`	The server source with all comments stripped (***Comments removed***)	`make`
`server-comments.c`	The server source code with comments.	`make -f Makefile-build-comments`
`testing.docx`	A document containing various tests.	-

`valgrind-ignore-ncurses.supp`	While developing the application I had not used valgrind test for memory leaks, I checks each function for any pointers that should be freed up and then note it down. After each function I made sure they were freed or streams like files were closed. When I finally got around to testing for any memory leaks I had noticed that I had numerous leaks coming from the ncurses library and created this suppression file to ignore all ncurses based detections.
	
	For more information from the ncurses documentation http://invisible-island.net/ncurses/ncurses.faq.html#config_leaks
		`valgrind --leak-check=full --show-leak-kinds=all --leak-resolution=med --track-origins=yes --suppressions=valgrind-ignore-ncurses.supp $1`
		
		`where $1 is the file being tested`

1.2. Software Setup and Compilation

1.2.1. Requirements / Prerequisites

This software has been tested on 2 distributions, Ubuntu and Linux Mint (with all packages installed). Although this software was developed on Ubuntu which already had ncurses installed and Linux Mint does not have ncurses installed by default, to compile this project it is required to be installed using the commands below.

Required Libraries: `ncurses-dev`
Required Software: `gcc, make`

To successfully install the software on a Debian based distribution that has apt-get then you can run this single command to get all the required libraries and software.

```
$ sudo apt-get update
$ sudo apt-get install gcc make ncurses-dev
```

After installing those packages, you can now compile the application

1.2.2. Software Compilation

The compilation of this project makes use of make files so you don't have to type any specific commands to do any actions unless you want to do specialised actions. There are two make files in the project, `Makefile` uses `client.c` and `server.c` which are the source files without commands. `Makefile-build-comments` uses `client-comments.c` and `server-comments.c` which are files that contain additional commentary. Around 500 lines of comments in the client file and about 100 in the server.

There will be 2 directories created after compilation, 1. bin – The output files, 2. Obj – The linker object files.

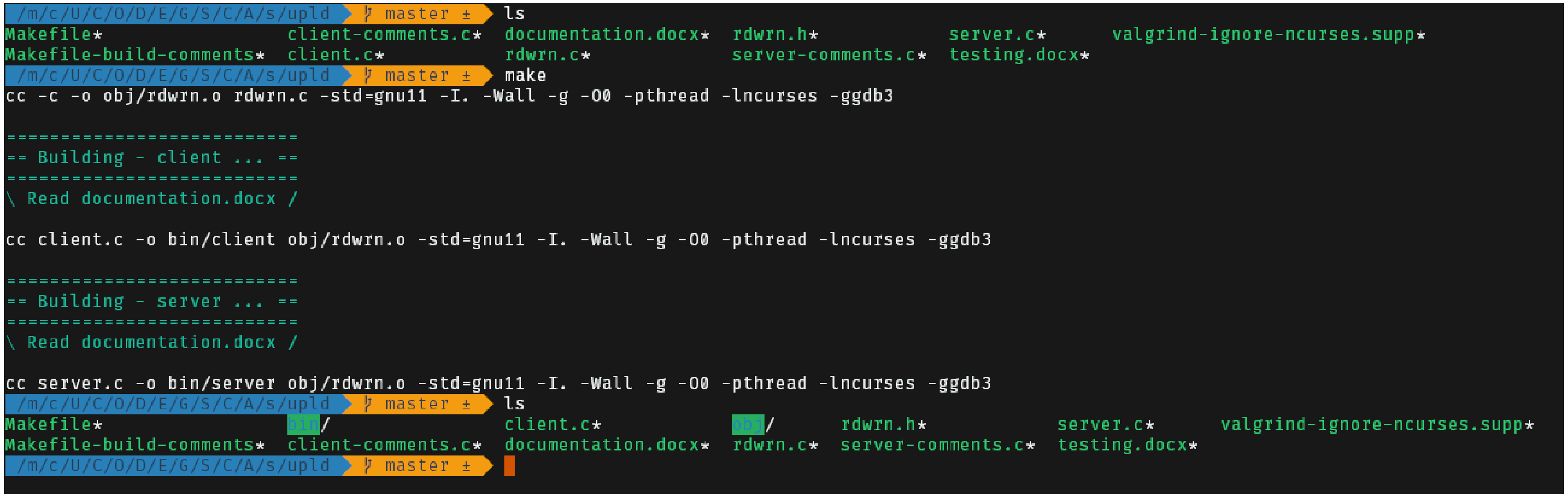
To compile the software without additional comments: `make`

To compile the software with the additional comments: `make -f Makefile-build-comments`

Each make file has 5 commands; `all, makedirs, client, server, clean`.

Command	*Description*
`all`	This will call makedirs then compile both the client and server.
`makedirs`	This will make the directories needed for compilation, bin and obj.
`client`	This command will compile the object file for rdwrn and compile client and place the executable into bin.
`server`	This command will compile the object file for rdwrn and then compile server and place its executable into bin.
`clean`	This command will delete the obj file and the files bin/client and bin/server. **THIS WILL NOT DELETE THE BIN DIRECTORY ONLY THE 2 SPECIFIED FILES. (Subfolders are untouched)**

A successful compilation should look like this:



```
 /m/c/U/C/O/D/E/G/S/C/A/s/upld   master ±  ls
Makefile*                 client-comments.c*  documentation.docx*  rdwrn.h*            server.c*      valgrind-ignore-ncurses.supp*
Makefile-build-comments*  client.c*           rdwrn.c*             server-comments.c*  testing.docx*
 /m/c/U/C/O/D/E/G/S/C/A/s/upld   master ±  make
cc -c -o obj/rdwrn.o rdwrn.c -std=gnu11 -I. -Wall -g -O0 -pthread -lncurses -ggdb3

===========================
== Building - client ... ==
===========================
\ Read documentation.docx /

cc client.c -o bin/client obj/rdwrn.o -std=gnu11 -I. -Wall -g -O0 -pthread -lncurses -ggdb3

===========================
== Building - server ... ==
===========================
\ Read documentation.docx /

cc server.c -o bin/server obj/rdwrn.o -std=gnu11 -I. -Wall -g -O0 -pthread -lncurses -ggdb3
 /m/c/U/C/O/D/E/G/S/C/A/s/upld   master ±  ls
Makefile*                 bin/                client.c*            obj/      rdwrn.h*            server.c*      valgrind-ignore-ncurses.supp*
Makefile-build-comments*  client-comments.c*  documentation.docx*  rdwrn.c*  server-comments.c*  testing.docx*
 /m/c/U/C/O/D/E/G/S/C/A/s/upld   master ± 
```

3. Software User Guide

Before reading this section go ahead and read Section 2 to learn how to compile the software, if you have already compiled the binar ies you can skip Section 2.

Please note currently the client only supports connections to localhost, if you want to change the destination address please modify `client.c` LN 572 and `client-comments.c` LN 772. This will allow you to connect to a different host.

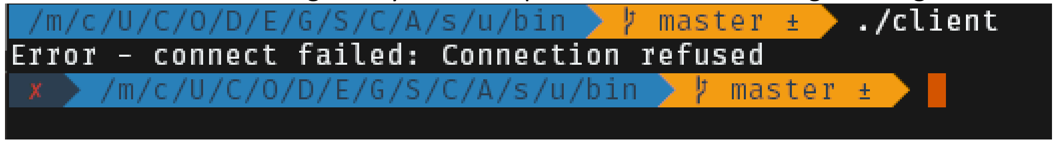
1.3.1. Simple usages

The software usage is created simple to remove any ambiguity. There are two executables provided after compilation. client and server. For the application to work as intended there needs to be a server running, this will use the **port 50031**. After the server is loaded and running the client can connect and start to communicate with the client.

The server is ready for client connections when you see output that looks like this:

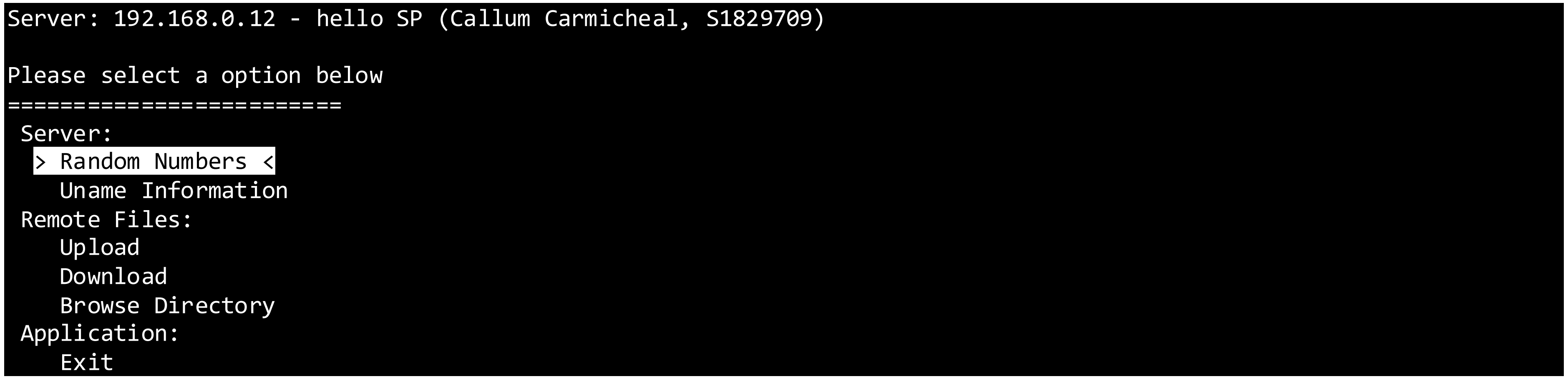
```
$ ./server
Waiting for incoming connections...
Waiting for a client to connect...
```

If the server is not running then you will be presented with a message stating the server is not running:



```
 /m/c/U/C/O/D/E/G/S/C/A/s/u/bin  master ±  ./client
Error - connect failed: Connection refused
 ✗  /m/c/U/C/O/D/E/G/S/C/A/s/u/bin  master ± 
```

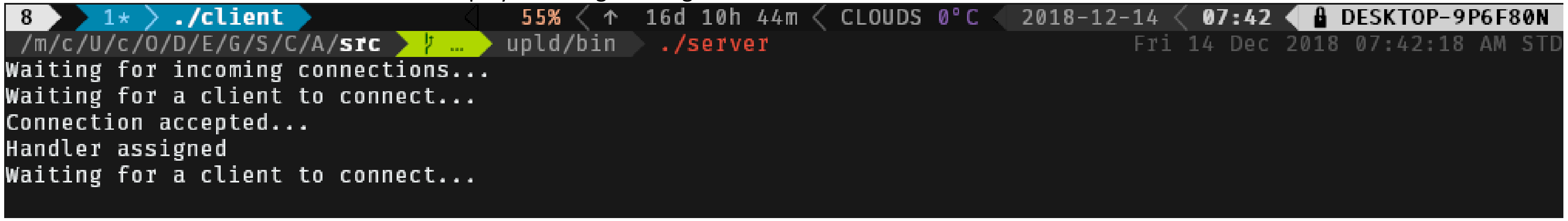
After that you will be able to execute the client executable and be presented with the interactive menu for the client:



```
Server: 192.168.0.12 - hello SP (Callum Carmicheal, S1829709)

Please select a option below
=========================
 Server:
  > Random Numbers <
    Uname Information
 Remote Files:
    Upload
    Download
    Browse Directory
 Application:
    Exit
```

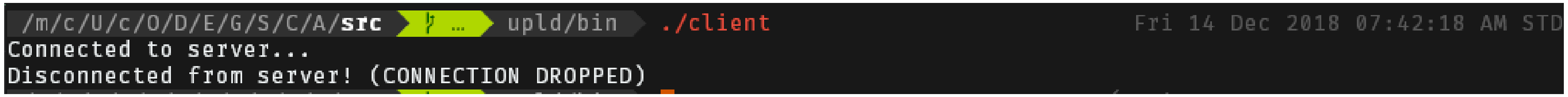
Please note that depending on your distribution, terminal multiplexer and colour support you might be have a blue background displayed instead of a black one. This was the case on Ubuntu where the background was blue but inside Linux Mint the background was black.

Once the client is connected the server will display a message stating the client has connected.



```
 8   1*  ./client        55%  ↑ 16d 10h 44m  CLOUDS 0°C  2018-12-14  07:42  DESKTOP-9P6F80N
 /m/c/U/c/O/D/E/G/S/C/A/src  ...  upld/bin  ./server       Fri 14 Dec 2018 07:42:18 AM STD
Waiting for incoming connections...
Waiting for a client to connect...
Connection accepted...
Handler assigned
Waiting for a client to connect...
```

Now that the client is connected all commands can be executed. The server supports multithreading so multiple clients can connect at once. Each command has error checking at almost every stage. If the server is closed the client will close and display a message.



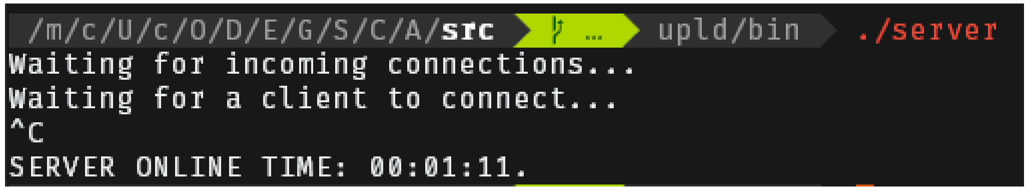
```
/m/c/U/c/O/D/E/G/S/C/A/src  ⎇ …  upld/bin  ./client                Fri 14 Dec 2018 07:42:18 AM STD
Connected to server...
Disconnected from server! (CONNECTION DROPPED)
```

1.3.2. Client Navigation and Features

Keys: The control set for the application is simple.

Command	*Description*
ENTER	Select the current highlighted menu item
Q or ESCAPE	Quit / Return without selection
DOWN ARROW	Go down in the menu
UP ARROW	Go up in the menu
SCROLLING	Scrolling up or Down will scroll the menu selection

Server Statistics: When the server is closing or receives a SIG INTERUPT (C-c) then the server will close and display the hours, minutes and seconds the server was online for.

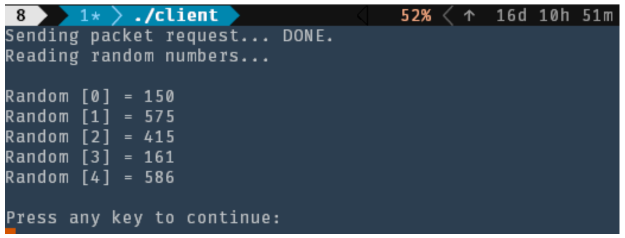


```
/m/c/U/c/O/D/E/G/S/C/A/src  ⎇ …  upld/bin  ./server
Waiting for incoming connections...
Waiting for a client to connect...
^C
SERVER ONLINE TIME: 00:01:11.
```

Server Fingerprinting: At the top of the client you will see information about the server, the first is the ip address. This is the address of either WIFI0 or ETH0. If it cannot locate WIFI0 then ETH0 is used instead. Followed by a welcome message with the author's name and student id.

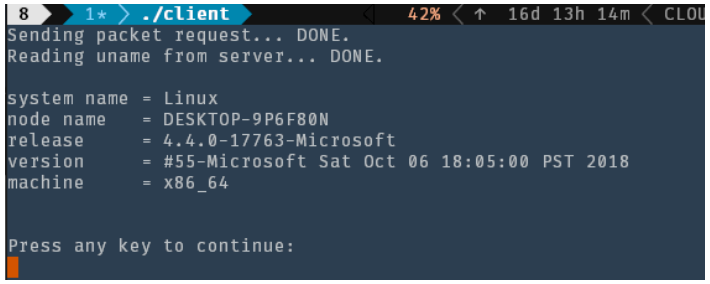
```
Server: 192.168.0.12 - hello SP (Callum Carmicheal, S1829709)
```

Random Numbers: The client can request an array of 5 random numbers ranging from 0 to 1000 from the server, these are then returned. The randomness is based of the internal clock of the computer so if there are subsequent requests in succession random numbers may not appear random. This is due to the seeding process and is normal.



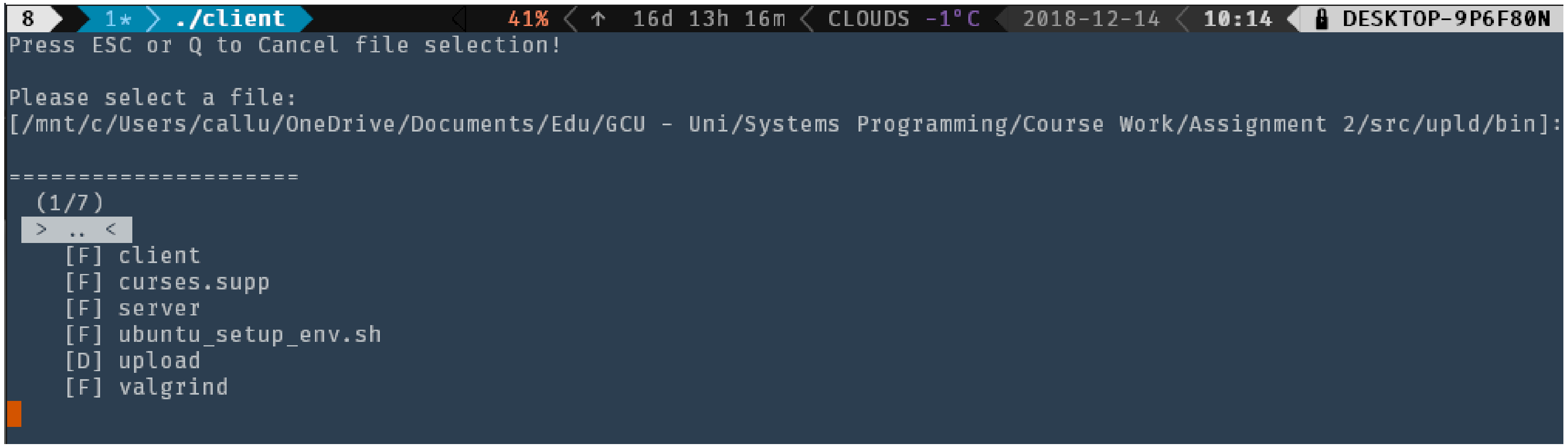
```
 8   1* > ./client                          52% < ↑ 16d 10h 51m
Sending packet request... DONE.
Reading random numbers...

Random [0] = 150
Random [1] = 575
Random [2] = 415
Random [3] = 161
Random [4] = 586

Press any key to continue:
```

Uname Information: This retrieves the uname information from the server and sends it to the client. If there are any errors on the server the client will receive a error code and then display it.

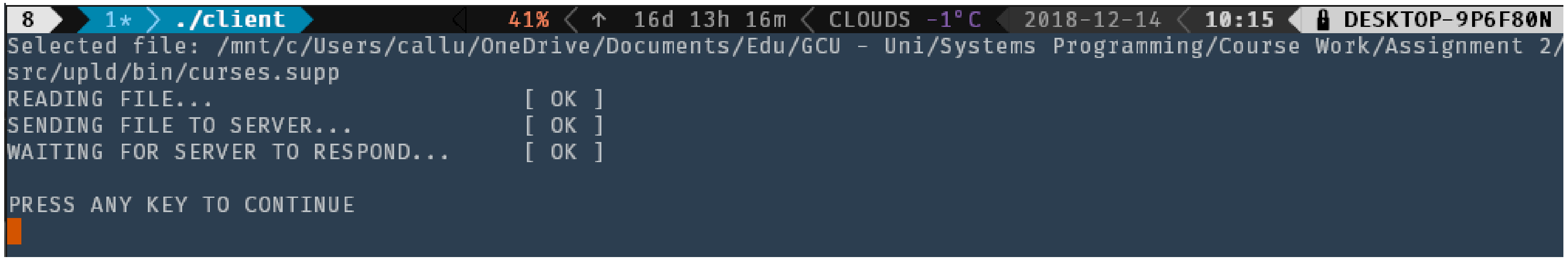


```
 8   1* > ./client                          42% < ↑ 16d 13h 14m < CLOU
Sending packet request... DONE.
Reading uname from server... DONE.

system name = Linux
node name   = DESKTOP-9P6F80N
release     = 4.4.0-17763-Microsoft
version     = #55-Microsoft Sat Oct 06 18:05:00 PST 2018
machine     = x86_64


Press any key to continue:
```

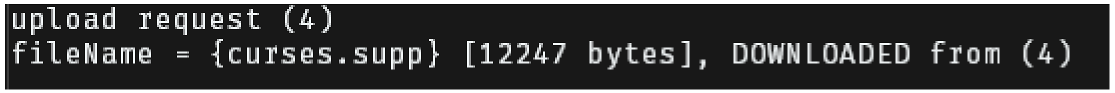
Remote – Upload: This section allows you to upload files from the current clients computer onto the server's computer. This features a interactive file browser, should you want to cancel this action just press Q or Escape.



```
 8 〉 1* 〉 ./client     41% 〈 ↑ 16d 13h 16m 〈 CLOUDS -1°C 〈 2018-12-14 〈 10:14 〈 DESKTOP-9P6F80N
Press ESC or Q to Cancel file selection!

Please select a file:
[/mnt/c/Users/callu/OneDrive/Documents/Edu/GCU - Uni/Systems Programming/Course Work/Assignment 2/src/upld/bin]:

====================
  (1/7)
 > .. <
    [F] client
    [F] curses.supp
    [F] server
    [F] ubuntu_setup_env.sh
    [D] upload
    [F] valgrind
```

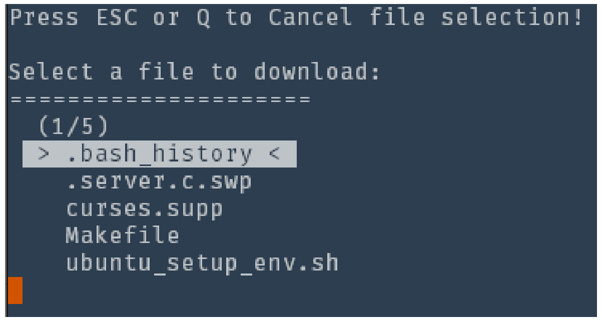


```
 8 〉 1* 〉 ./client     41% 〈 ↑ 16d 13h 16m 〈 CLOUDS -1°C 〈 2018-12-14 〈 10:15 〈 DESKTOP-9P6F80N
Selected file: /mnt/c/Users/callu/OneDrive/Documents/Edu/GCU - Uni/Systems Programming/Course Work/Assignment 2/
src/upld/bin/curses.supp
READING FILE...                      [ OK ]
SENDING FILE TO SERVER...            [ OK ]
WAITING FOR SERVER TO RESPOND...     [ OK ]

PRESS ANY KEY TO CONTINUE
```

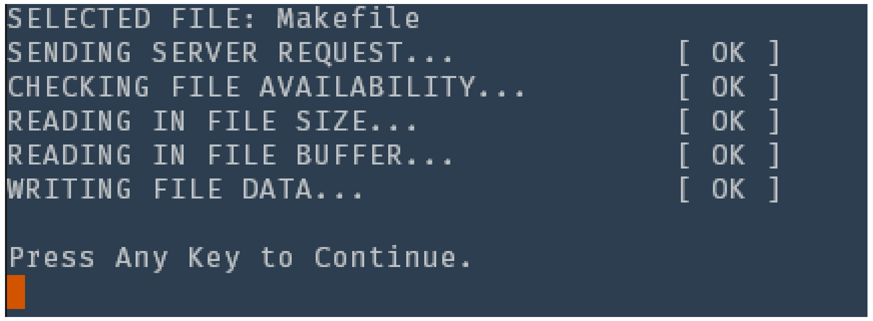


```
upload request (4)
fileName = {curses.supp} [12247 bytes], DOWNLOADED from (4)
```

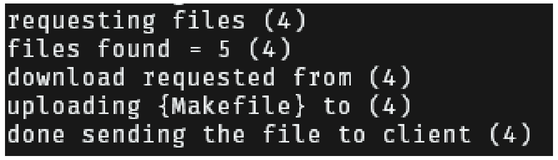
Remote – Download: Download files from the upload directory to the current directory on the client. If there are no files on the server there will be a error message displayed. If there are files an file selection will be shown.



```
Press ESC or Q to Cancel file selection!

Select a file to download:
=====================
  (1/5)
  > .bash_history <
    .server.c.swp
    curses.supp
    Makefile
    ubuntu_setup_env.sh
```

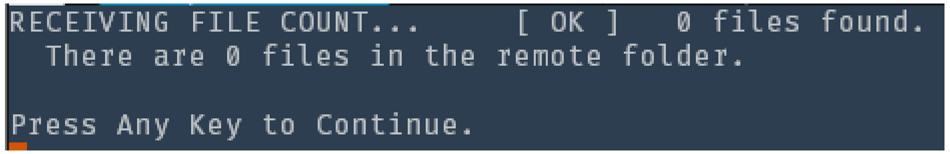


```
SELECTED FILE: Makefile
SENDING SERVER REQUEST...                    [ OK ]
CHECKING FILE AVAILABILITY...                [ OK ]
READING IN FILE SIZE...                      [ OK ]
READING IN FILE BUFFER...                    [ OK ]
WRITING FILE DATA...                         [ OK ]

Press Any Key to Continue.
```



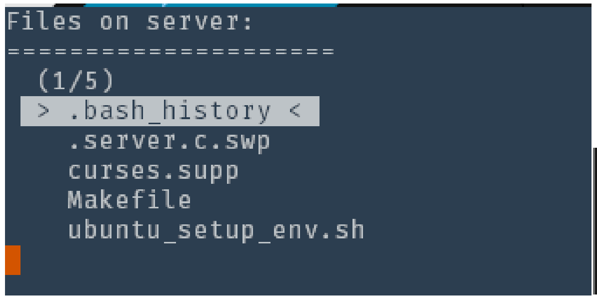
```
requesting files (4)
files found = 5 (4)
download requested from (4)
uploading {Makefile} to (4)
done sending the file to client (4)
```

If there are no files, then this is displayed:

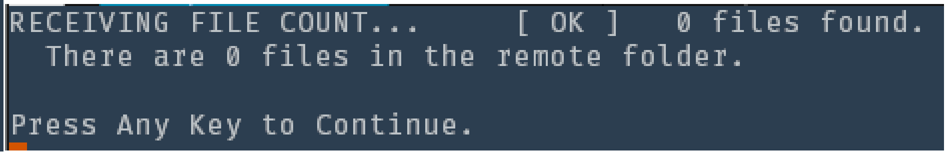


```
RECEIVING FILE COUNT...      [ OK ]   0 files found.
  There are 0 files in the remote folder.

Press Any Key to Continue.
```

Remote – Browse: Browse the files inside the upload directory. If there are none a error is displayed.



```
Files on server:
=====================
  (1/5)
  > .bash_history <
    .server.c.swp
    curses.supp
    Makefile
    ubuntu_setup_env.sh
```



```
RECEIVING FILE COUNT...      [ OK ]   0 files found.
  There are 0 files in the remote folder.

Press Any Key to Continue.
```

2. Systems Programming – Socket Application Testing

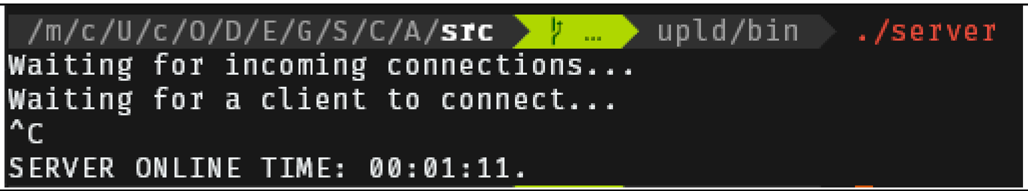
Student: Callum Carmicheal, S1829709

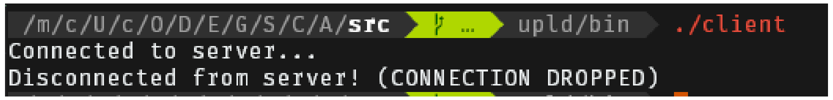
2.1. Test Cases

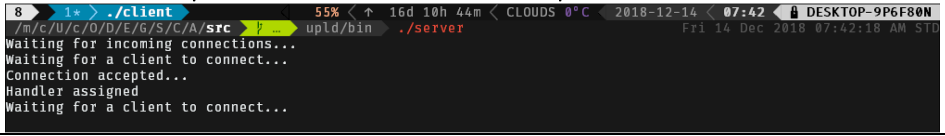
Test Number	Purpose of Test
1.	Server SIG INT / Display Time on closing.
2.	Closing server prematurely to simulate a network disconnection or server crash.
3.	Accepting a socket.
4.	Receiving a random number from the server.
5.	Opening the client when the server is turned off.
6.	Receiving the servers uname information.
7.	Uploading a remote file to the server from the client .
8.	Uploading a remote file to the server from the client without the uploads directory existing.
9.	Downloading a file from the server when the uploads directory does not exist.
10.	Browsing the uploads directory from the server when the uploads directory does not exist.
11.	Browsing the uploads directory.
12.	Downloading a file from the server to the cwd.
13.	Valgrind – Client – Searching for any memory leaks.
14.	Valgrind – Server – Searching for any memory leaks.
15.	Menu System – Viewing a set of items where the list is larger then the visible viewport.

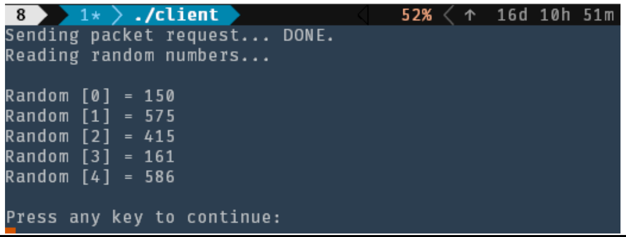
2.2. Testing Data

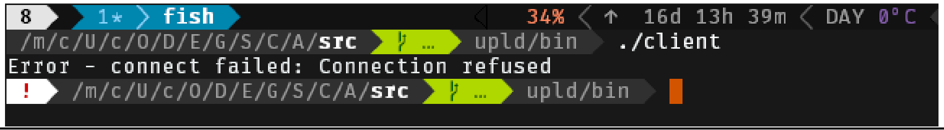
Systems Programming Coursework 2, Trimester A			
Testing Chart			
Test Number	Command	Expected Result	Actual Result

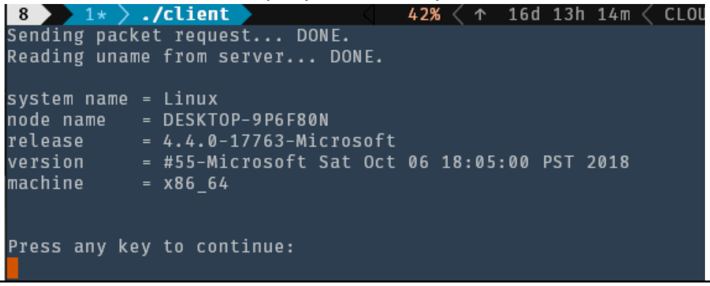
1.	`./server`
	
	After some time
	
	C-c
		The server to close with the online time being displayed.	
			`/m/c/U/c/O/D/E/G/S/C/A/src ᚴ … upld/bin ./server`
			`Waiting for incoming connections...`
			`Waiting for a client to connect...`
			`^C`
			`SERVER ONLINE TIME: 00:01:11.`

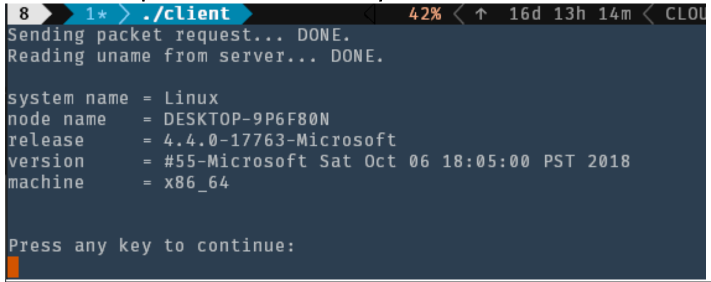
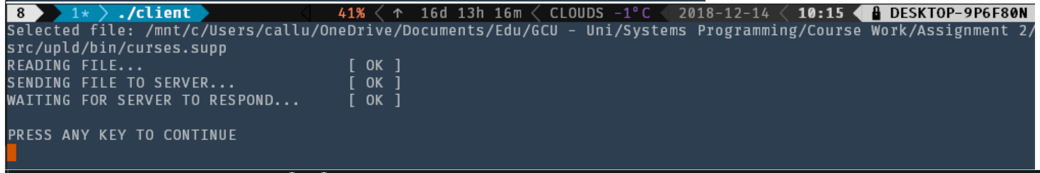
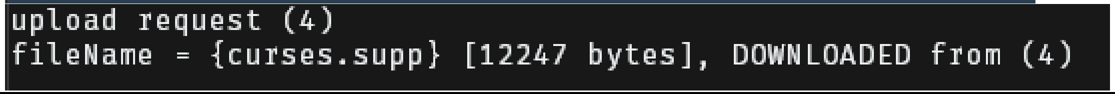
2.	`./server`
	`./client`
	
	After some time
	
	C-c on the server
		The client to disconnect instantly.	The client does not disconnect for some reason, I believe this is something to do with the client not freeing up the socket or it's the system taking some time to free up the socket. Either way the client can read about 10 to 20 bytes before the client closes with this message:
			
			`/m/c/U/c/O/D/E/G/S/C/A/src ᚴ … upld/bin ./client`
			`Connected to server...`
			`Disconnected from server! (CONNECTION DROPPED)`

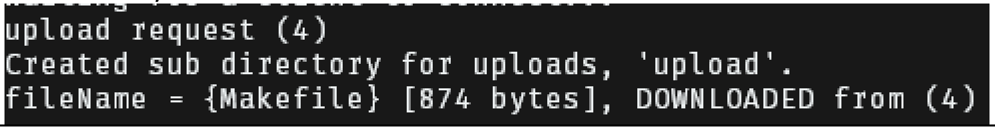
3.	`./server`
	`./client`
		The server to accept the socket and the client to open	The server accepts the socket and the client opens.
			
			`8 1* ./client 55% ↑ 16d 10h 44m CLOUDS 0°C 2018-12-14 07:42 DESKTOP-9P6F80N`
			`/m/c/U/c/O/D/E/G/S/C/A/src ᚴ … upld/bin ./server Fri 14 Dec 2018 07:42:18 AM STD`
			`Waiting for incoming connections...`
			`Waiting for a client to connect...`
			`Connection accepted...`
			`Handler assigned`
			`Waiting for a client to connect...`

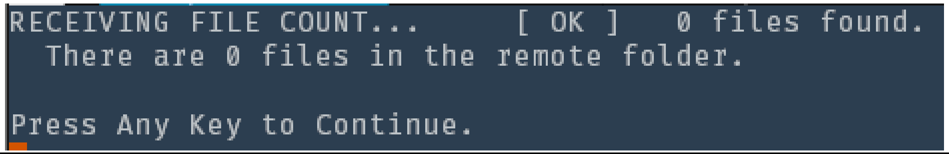
4.	`./client`	The number to be random	The number is randomised although if the requests are send in quick succession because the seeding and time based randomness the number will not be random, its usually around every 500ms to 1s the number is randomised. This is because its not a true random number, it's a psuedo random number.
			
			`8 1* ./client 52% ↑ 16d 10h 51m`
			`Sending packet request... DONE.`
			`Reading random numbers...`
			
			`Random [0] = 150`
			`Random [1] = 575`
			`Random [2] = 415`
			`Random [3] = 161`
			`Random [4] = 586`
			
			`Press any key to continue:`

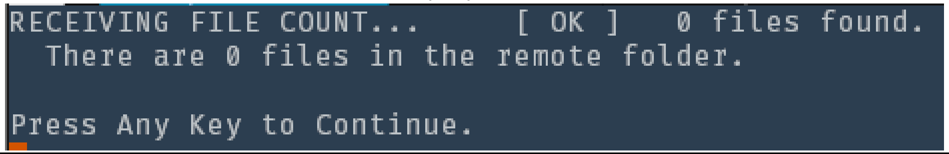
5.	`./client`	The client should show a error message	A error message is displayed saying the server is offline or it cannot connect.
			
			`8 1* fish 34% ↑ 16d 13h 39m DAY 0°C`
			`/m/c/U/c/O/D/E/G/S/C/A/src ᚴ … upld/bin ./client`
			`Error - connect failed: Connection refused`
			`! /m/c/U/c/O/D/E/G/S/C/A/src ᚴ … upld/bin`

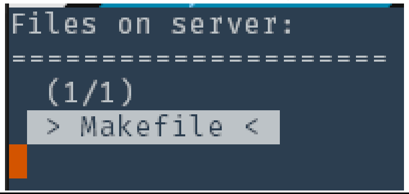
6.	`./client`	The uname information is received and displayed	The information is dispalyed corectly
			
			`8 1* ./client 42% ↑ 16d 13h 14m CLOU`
			`Sending packet request... DONE.`
			`Reading uname from server... DONE.`
			
			`system name = Linux`
			`node name   = DESKTOP-9P6F80N`
			`release     = 4.4.0-17763-Microsoft`
			`version     = #55-Microsoft Sat Oct 06 18:05:00 PST 2018`
			`machine     = x86_64`
			
			`Press any key to continue:`

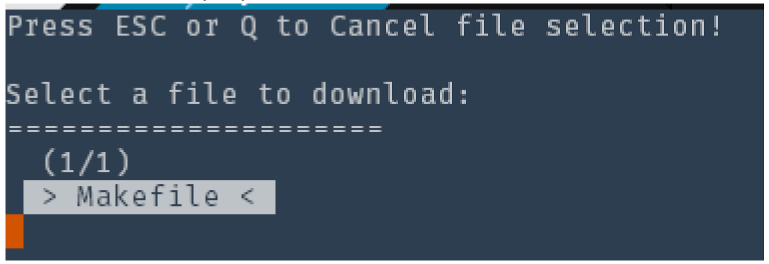
7.	`./client`	The file is uploaded without any errors	The file is uploaded without any errors
			
			`8 1* ./client 42% ↑ 16d 13h 14m CLOU`
			`Sending packet request... DONE.`
			`Reading uname from server... DONE.`
			
			`system name = Linux`
			`node name   = DESKTOP-9P6F80N`
			`release     = 4.4.0-17763-Microsoft`
			`version     = #55-Microsoft Sat Oct 06 18:05:00 PST 2018`
			`machine     = x86_64`
			
			`Press any key to continue:`
			
			`8 1* ./client 41% ↑ 16d 13h 16m CLOUDS -1°C 2018-12-14 10:15 DESKTOP-9P6F80N`
			`Selected file: /mnt/c/Users/callu/OneDrive/Documents/Edu/GCU - Uni/Systems Programming/Course Work/Assignment 2/`
			`src/upld/bin/curses.supp`
			`READING FILE...                [ OK ]`
			`SENDING FILE TO SERVER...       [ OK ]`
			`WAITING FOR SERVER TO RESPOND... [ OK ]`
			
			`PRESS ANY KEY TO CONTINUE`
			
			`upload request (4)`
			`fileName = {curses.supp} [12247 bytes], DOWNLOADED from (4)`

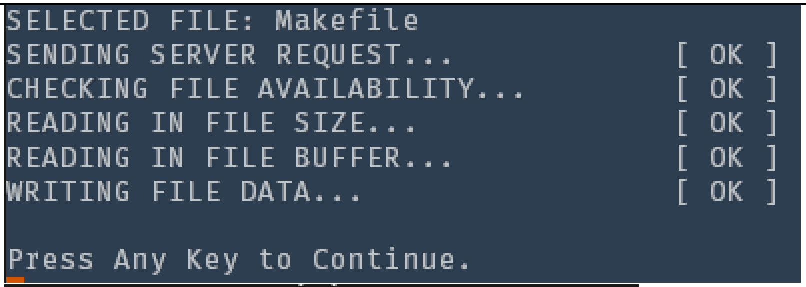
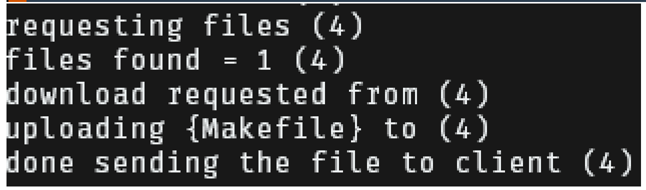
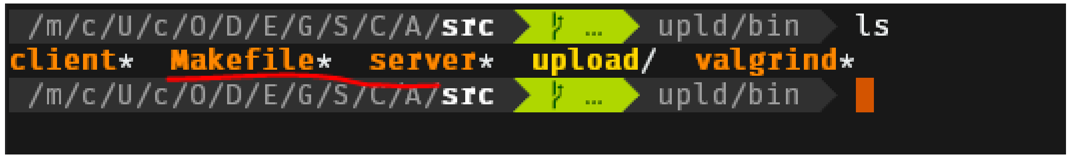
8.	`./server`	The directory is created automatically and then the file is uploaded.	The directory and file are created.
			
			`upload request (4)`
			`Created sub directory for uploads, 'upload'.`
			`fileName = {Makefile} [874 bytes], DOWNLOADED from (4)`

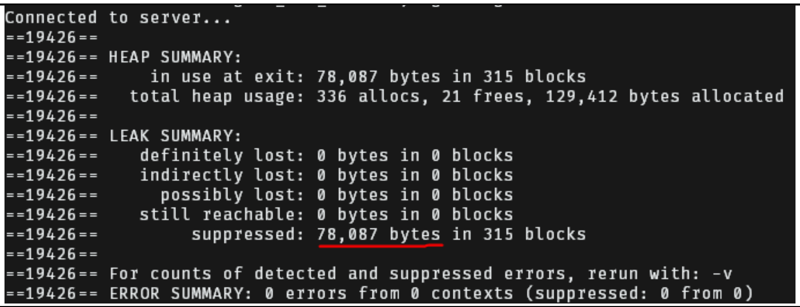
9.	`./client`	A error message is displayed instead of the menu	There are no files so the error is displayed  `RECEIVING FILE COUNT...       [ OK ]    0 files found.`
			`  There are 0 files in the remote folder.`
			
			`Press Any Key to Continue.`

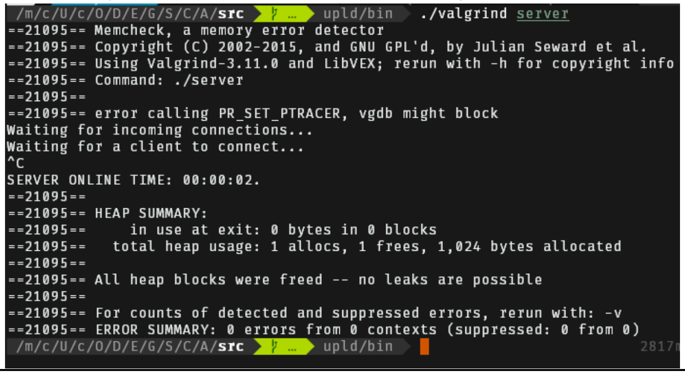
10.	`./client`	A error message is displayed instead of the menu	There are no files so the error is displayed  `RECEIVING FILE COUNT...       [ OK ]    0 files found.`
			`  There are 0 files in the remote folder.`
			
			`Press Any Key to Continue.`

11.	`./client`	The files in the server are displayed	The files in the directory are received and rendered.  `Files on server:`
			`=====================`
			`  (1/1)`
			` > Makefile <`

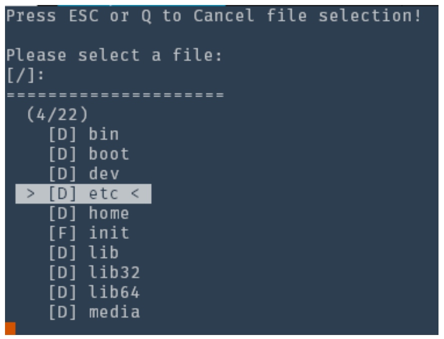
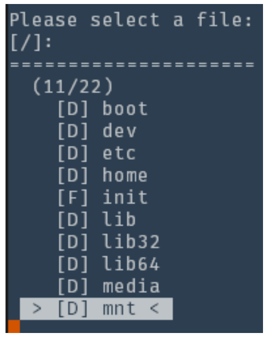
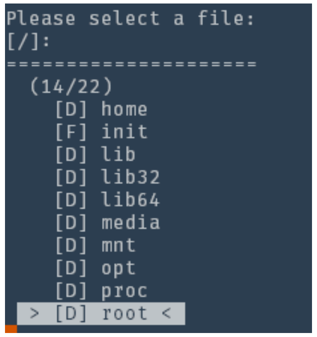
12.	`./client`	The file is downloaded to the cwd of the client	The file list is displayed and “Makefile” is selected.  `Press ESC or Q to Cancel file selection!`
			
			`Select a file to download:`
			`=====================`
			`  (1/1)`
			` > Makefile <`
			
			The file is then downloaded

			 `SELECTED FILE: Makefile` `SENDING SERVER REQUEST... [ OK ]` `CHECKING FILE AVAILABILITY... [ OK ]` `READING IN FILE SIZE... [ OK ]` `READING IN FILE BUFFER... [ OK ]` `WRITING FILE DATA... [ OK ]` `Press Any Key to Continue.`  `requesting files (4)` `files found = 1 (4)` `download requested from (4)` `uploading {Makefile} to (4)` `done sending the file to client (4)` The file shows in cwd  `/m/c/U/c/O/D/E/G/S/C/A/src upld/bin ls` `client* Makefile* server* upload/ valgrind*` `/m/c/U/c/O/D/E/G/S/C/A/src upld/bin`
13.	`valgrind --leak-check=full --show-leak-kinds=all --leak-resolution=med --track-origins=yes ./client`	There should be no memory leaks associated with the client application.	Upon first running of valgrind I noticed I had 40 memory leaks and had decided to bring a sledge hammer to the source code, upon adding additional arguments to valgrind to see what was going on, I had noticed most of these leaks, infact all of them were caused by ncurses. So I add created a valgrind supression file labelled `valgrind-ignore-ncurses.supp` After excluding all ncurses leaks I had none.

			
			`Connected to server...`
			`==19426==`
			`==19426== HEAP SUMMARY:`
			`==19426==     in use at exit: 78,087 bytes in 315 blocks`
			`==19426==   total heap usage: 336 allocs, 21 frees, 129,412 bytes allocated`
			`==19426==`
			`==19426== LEAK SUMMARY:`
			`==19426==    definitely lost: 0 bytes in 0 blocks`
			`==19426==    indirectly lost: 0 bytes in 0 blocks`
			`==19426==      possibly lost: 0 bytes in 0 blocks`
			`==19426==    still reachable: 0 bytes in 0 blocks`
			`==19426==         suppressed: 78,087 bytes in 315 blocks`
			`==19426==`
			`==19426== For counts of detected and suppressed errors, rerun with: -v`
			`==19426== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)`

14.	`valgrind --leak-check=full --show-leak-kinds=all --leak-resolution=med --track-origins=yes ./server`	There should be no memory leaks associated with the server.	During development I had found 3 memory leaks, most of these were associated with strings that had been allocated (malloc) during the file upload / download process, these were not cleaned up during failed read or writes to the client so the socket was closed but the memory was not released.
			
			Currently the server does not have any leaks.
			
			`/m/c/U/c/O/D/E/G/S/C/A/src  ...  upld/bin  ./valgrind server`
			`==21095== Memcheck, a memory error detector`
			`==21095== Copyright (C) 2002-2015, and GNU GPL'd, by Julian Seward et al.`
			`==21095== Using Valgrind-3.11.0 and LibVEX; rerun with -h for copyright info`
			`==21095== Command: ./server`
			`==21095==`
			`==21095== error calling PR_SET_PTRACER, vgdb might block`
			`Waiting for incoming connections...`
			`Waiting for a client to connect...`
			`^C`
			`SERVER ONLINE TIME: 00:00:02.`
			`==21095==`
			`==21095== HEAP SUMMARY:`
			`==21095==     in use at exit: 0 bytes in 0 blocks`
			`==21095==   total heap usage: 1 allocs, 1 frees, 1,024 bytes allocated`
			`==21095==`
			`==21095== All heap blocks were freed -- no leaks are possible`
			`==21095==`
			`==21095== For counts of detected and suppressed errors, rerun with: -v`
			`==21095== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)`
			`/m/c/U/c/O/D/E/G/S/C/A/src  ...  upld/bin  2817`

15.	`./client`	The user should be able to scroll them menu even if it exceeds the viewport. This includes viewing labels (non-buttons).	To show this functionality I will be displaying the root folder which exceeds 10 items (which is current the amount of items limited to be rendered): There are 22 folders and if I scroll I am able to expand the list:

```
Press ESC or Q to Cancel file selection!

Please select a file:
[/]:
=====================
  (4/22)
    [D] bin
    [D] boot
    [D] dev
  > [D] etc <
    [D] home
    [F] init
    [D] lib
    [D] lib32
    [D] lib64
    [D] media
```

```
Please select a file:
[/]:
=====================
  (11/22)
    [D] boot
    [D] dev
    [D] etc
    [D] home
    [F] init
    [D] lib
    [D] lib32
    [D] lib64
    [D] media
  > [D] mnt <
```

```
Please select a file:
[/]:
=====================
  (14/22)
    [D] home
    [F] init
    [D] lib
    [D] lib32
    [D] lib64
    [D] media
    [D] mnt
    [D] opt
    [D] proc
  > [D] root <
```