# Word Guess

This assignment gives the programmer experience with forking to a create child process. Named pipes are used to communicate between the server process and a client process. This assignment will specify what the parent process and the child process requirements are in the program. It also describes what the client must do to communicate with the server.

The program will play a word guessing game between a game background server and a game client. The server provides a random guess word profile to the client. The client guesses letters that are written to the server that fills in any matches in the guess word slots. The guess word is then written back to the client. The client gets 12 letter guesses for a word.

**General Program Requirements**

Use global constants for use by the program:

All arrays that require a size declarator should be of size declarator of 100.

You must use try-catch and throws for all Unix APIs, especially named pipe commands, using the LineInfo.h exception capability that has been used in all the programs in this course.

On cs1 Linux system (the target platform), the mkfifo API returns -1 even if the operation is successful.  Therefore, do not use the LineInfo.h exception handler with throws for the mkfifo API.

Use the standard file operations for the named pipes – open(), read(), write(), close() … etc.
Do not use FILE and the f_ file commands (file descriptors) for any file operations.

Use C++ (g++) features (iostream, string… etc.).
Build using g++ -std=c++11.

The names of the source programs and associated execution files are:

gserver.cpp
gserver

gclient.cpp
gclient

**Program Command Line**

To run the game:

./gserver words.txt &
./gclient

The words.txt source will contain words that are used to pick out a random word.

The server waits for a client connection across a known named pipe that is known to both client and server.

When the server and client is done running one game, both the client and server must exit.

All created named pipes must be in the development directory.

The programs must close and unlink all created pipes when done with a agme.

# Word Guess

**Words File**

The words file which is an argument to the server execution file is located at:
~smd013000/courses/3377/assigns/05/words.txt

**Vector<string>**

The words from the file must be read into a vector using the copy library method.

**Named Pipes**

There will be three pipes used for each game:

1. A server created known named pipe, known by server and client, used at the beginning to establish private named pipes used by the server and by the client for messaging.

2. A client created unique private named pipe for server write to client reads.

3. A server created unique private named pipe for server write to client reads.

**gserver process**

server reads into a vector<string> from a file name provided by the argv[1].

server creates, then opens for reading a known named pipe for reading client requester message.

server reads in on the known pipe name a client message.

This client message contains the client created unique pipe name for server writes and client reads.

server is done with the requester known named pipe, so this named pipe is closed and unlinked.

server creates, then opens for writing the previously sent client created unique named pipe.

server creates a random word from the vector.

server creates a corresponding guess word made up of empty slots using dashes (-) to represent the empty letter slots.

server increments the client try count.

server forks()

**child process**

child process creates a string of the client try count

child process writes out this try count string to client

child process sleeps for 2 seconds to wait for client and send a read

child process writes out the random word string to client

child process creates a unique named pipe name for child server reads of client writes.

child process writes this unique pipe name out to the client.

child process creates, then opens for reading the child created unique named pipe.

# Word Guess

loop (game)

   child process writes guess word (with blanks) to client

   child process reads guess letter from client

   child process fills slots in guess word with any sent letter matches using random word index

loop end (game)

loop end (server)

close and unlink any named pipes

# Word Guess

**gclient process**

client creates a unique named pipe name string for server writes to child reads

client creates, then opens for writing, the known named pipe for client writing to server reading requests.

client writes to known named pipe and sends as a message that contains the client created unique pipe name string for server writes to child reads

client creates, then opens for reading, the named pipe string that was sent to the server, for server writes to child reads

client reads the server writes to child named pipe to get the client try number string message

client reads the server writes to child named pipe to get the random word string message

client reads the server writes to child named pipe to get the server read from client write string message

client creates, then opens for writes, the named pipe string that was sent from the server, for server reads to child writes

client sets number of tries to 0

client announces game start

loop start (game)

- client reads server write to client red named pipe a guess word
- client determines if guess word is same as the random word, if so it declares a win, then breaks from loop
- client determines if number of maximum tries has been exceeded, if so states exceeded num of tries, the random word, and then breaks from loop
- client display status and asks for a guess letter (see sample run at end of this document)
- client writes to server read from client write named pipe the guess letter
- client increments number of tries

loop end (game)

Client closes and unlinks from all named pipes

# Word Guess

**Activities File**

After done designing and testing, capture the run results in an activities text file named activities.txt.

This activities.txt file must contain a successful guess and an unsuccessful guess.

Use the copy and paste to local editor text file for creating the wordguessactivities.txt file.

Do not use script or terminal session to create the wordguessactivities.txt file.

See the sample run below.

**Submittal**

Put in a zip folder that you name:

gserver.cpp
gclient.cpp
activities.txt

Due to its size, do not put in the words.txt file in the zip folder, the tag will provide it when grading your programs.

Submit the .zip folder to blackboard for this assignment.

Sample Run

```
{cslinux1:~/courses/3377/assigns/05} ./gserver words.txt &
{cslinux1:~/courses/3377/assigns/05} ./gclient
fork

Game Start
You have 12 letter guesses to win

Client : 1
No of tries : 0
(Guess) Enter a letter in the word : ----------

a

Client : 1
No of tries : 1
(Guess) Enter a letter in the word : ---a------

e

Client : 1
No of tries : 2
(Guess) Enter a letter in the word : ---a------

i

Client : 1
No of tries : 3
(Guess) Enter a letter in the word : ---a------

o
```

# Word Guess

```
Client : 1
No of tries : 4
(Guess) Enter a letter in the word : ---a-o-o--

u

Client : 1
No of tries : 5
(Guess) Enter a letter in the word : u--a-o-ou-

n

Client : 1
No of tries : 6
(Guess) Enter a letter in the word : un-a-o-ou-

t

Client : 1
No of tries : 7
(Guess) Enter a letter in the word : un-a-o-ou-

s
```

# Word Guess

```
Client : 1
No of tries : 8
(Guess) Enter a letter in the word : un-a-o-ous

d

Client : 1
No of tries : 9
(Guess) Enter a letter in the word : un-a-o-ous

r

Client : 1
No of tries : 10
(Guess) Enter a letter in the word : un-a-orous

m

Client : 1
No of tries : 11
(Guess) Enter a letter in the word : un-a-orous

k

Client : 1
No of tries : 12
(Guess) Enter a letter in the word : un-a-orous

h


Out of tries : 12 allowed
The word is: unvalorous

{cslinux1:~/courses/3377/assigns/05}

{cslinux1:~/courses/3377/assigns/05} ./gserver words.txt &
[1] 13788
{cslinux1:~/courses/3377/assigns/05} ./gclient


Game Start
You have 12 letter guesses to win

Client : 1
No of tries : 0
(Guess) Enter a letter in the word : -----

a

Client : 1
No of tries : 1
(Guess) Enter a letter in the word : -----

e
```

# Word Guess

```
Client : 1
No of tries : 2
(Guess) Enter a letter in the word : -----

i

Client : 1
No of tries : 3
(Guess) Enter a letter in the word : i---

s

issis


You Win!

{cslinux1:~/courses/3377/assigns/05}
```