Client

1

Generated by Doxygen 1.8.14

Contents

Chapter 1

File Index

1.1 File List

Here is a list of all files with brief descriptions:

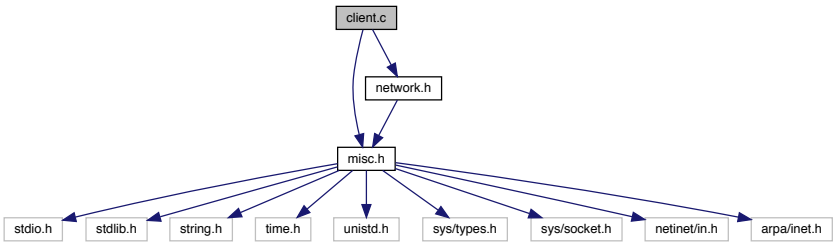
Chapter 2

File Documentation

2.1 client.c File Reference

```
#include "misc.h"
#include "network.h"
```

Include dependency graph for client.c:



Macros

- #define PORT 80

Functions

- int main (int argc, char ∗argv[ ])

2.1.1 Macro Definition Documentation

2.1.1.1 PORT

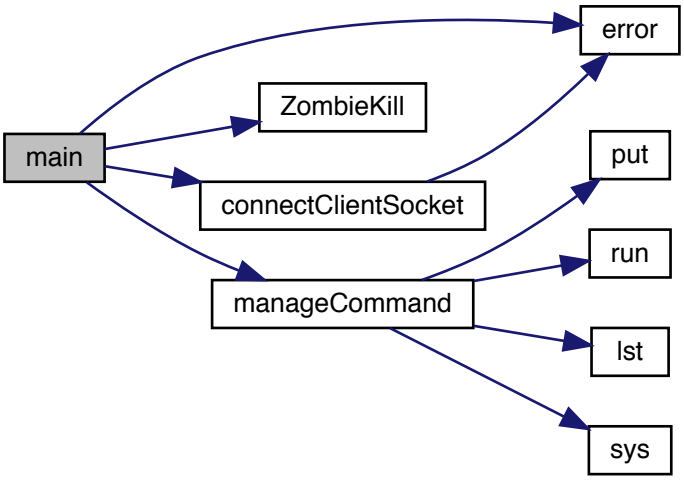
```
#define PORT 80
```

2.1.2 Function Documentation

2.1.2.1 main()

```
int main (
            int argc,
            char * argv[] )
```

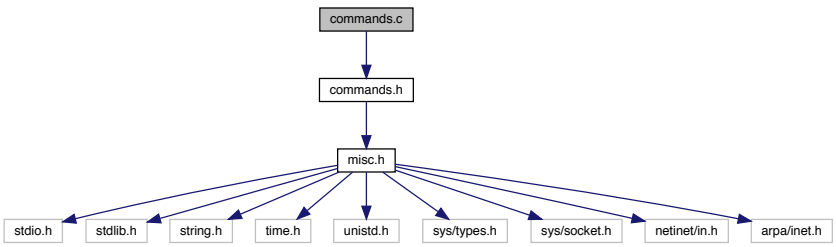
Loops waiting for user input then forks and deals with command. Here is the call graph for this function:



2.2 commands.c File Reference

```
#include "commands.h"
```

Include dependency graph for commands.c:



Functions

- int put (V_SOCKET serverSocket, char args[MAX_ARGS][DEFAULT_BUFLEN], int argsCount)
- int get (V_SOCKET serverSocket, char args[MAX_ARGS][DEFAULT_BUFLEN], int argsCount)
- int run (V_SOCKET serverSocket, char args[MAX_ARGS][DEFAULT_BUFLEN], int argsCount)
- int lst (V_SOCKET serverSocket, char args[MAX_ARGS][DEFAULT_BUFLEN], int argsCount)
- int sys (V_SOCKET serverSocket, char args[MAX_ARGS][DEFAULT_BUFLEN], int argsCount)

2.2.1 Function Documentation

2.2.1.1 get()

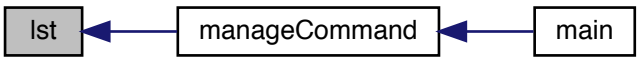
```
int get (
            V_SOCKET serverSocket,
            char args[MAX_ARGS][DEFAULT_BUFLEN],
            int argsCount )
```

Validates command then sends command to server. Then waits for "ready" or "error". if ready Receive 40 lines and display to user waiting for input to then display more.

2.2.1.2 lst()

```
int lst (
            V_SOCKET serverSocket,
            char args[MAX_ARGS][DEFAULT_BUFLEN],
            int argsCount )
```

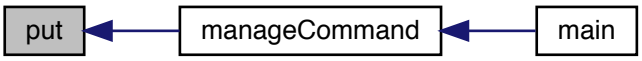
Validates command then sends command to server. Then waits for "ready" or "error". if ready Receive 40 lines and display to user waiting for input to then display more. Here is the caller graph for this function:



2.2.1.3 put()

```
int put (
            V_SOCKET serverSocket,
            char args[MAX_ARGS][DEFAULT_BUFLEN],
            int argsCount )
```

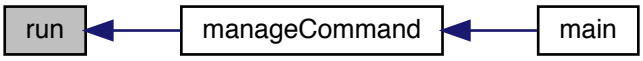
Validates command then sends command to server. Then waits for "ready" or "error". if ready then sends all files content to server. Here is the caller graph for this function:



2.2.1.4 run()

```
int run (
            V_SOCKET serverSocket,
            char args[MAX_ARGS][DEFAULT_BUFLEN],
            int argsCount )
```

Validates command then sends command to server. Then waits for "ready" or "error". If ready either displays return or writes to file given. Here is the caller graph for this function:



2.2.1.5 sys()

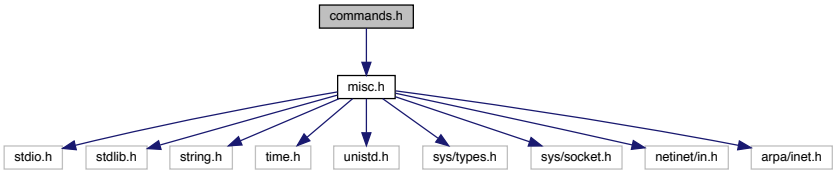
```
int sys (
            V_SOCKET serverSocket,
            char args[MAX_ARGS][DEFAULT_BUFLEN],
            int argsCount )
```

Sends command to server then displays the response if no error. Here is the caller graph for this function:

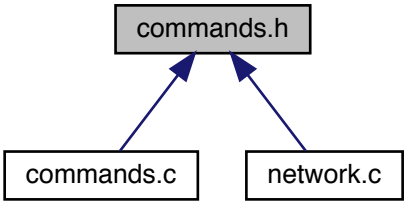
2.3 commands.h File Reference

```
#include "misc.h"
```

Include dependency graph for commands.h:



This graph shows which files directly or indirectly include this file:



Functions

- int put (V_SOCKET serverSocket, char args[MAX_ARGS][DEFAULT_BUFLEN], int argsCount)
- int get (V_SOCKET serverSocket, char args[MAX_ARGS][DEFAULT_BUFLEN], int argsCount)
- int run (V_SOCKET serverSocket, char args[MAX_ARGS][DEFAULT_BUFLEN], int argsCount)
- int lst (V_SOCKET serverSocket, char args[MAX_ARGS][DEFAULT_BUFLEN], int argsCount)
- int sys (V_SOCKET serverSocket, char args[MAX_ARGS][DEFAULT_BUFLEN], int argsCount)

2.3.1 Function Documentation

2.3.1.1 get()

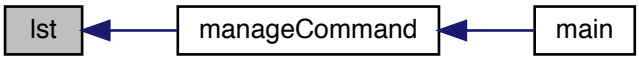
```
int get (
            V_SOCKET serverSocket,
            char args[MAX_ARGS][DEFAULT_BUFLEN],
            int argsCount )
```

Validates command then sends command to server. Then waits for "ready" or "error". if ready Receive 40 lines and display to user waiting for input to then display more.

2.3.1.2 lst()

```
int lst (
            V_SOCKET serverSocket,
            char args[MAX_ARGS][DEFAULT_BUFLEN],
            int argsCount )
```

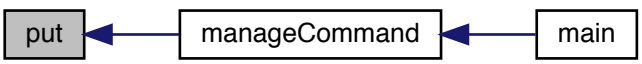
Validates command then sends command to server. Then waits for "ready" or "error". if ready Receive 40 lines and display to user waiting for input to then display more. Here is the caller graph for this function:



2.3.1.3 put()

```
int put (
            V_SOCKET serverSocket,
            char args[MAX_ARGS][DEFAULT_BUFLEN],
            int argsCount )
```

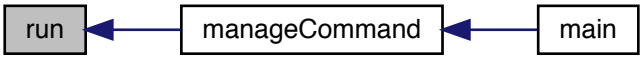
Validates command then sends command to server. Then waits for "ready" or "error". if ready then sends all files content to server. Here is the caller graph for this function:



2.3.1.4 run()

```
int run (
            V_SOCKET serverSocket,
            char args[MAX_ARGS][DEFAULT_BUFLEN],
            int argsCount )
```

Validates command then sends command to server. Then waits for "ready" or "error". If ready either displays return or writes to file given. Here is the caller graph for this function:



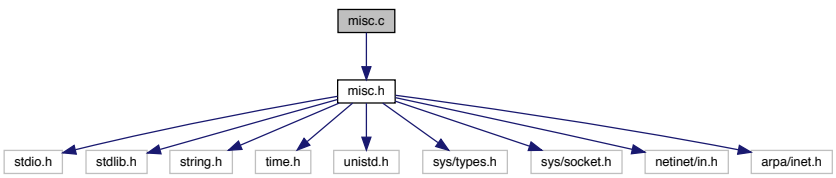
2.3.1.5 sys()

```
int sys (
            V_SOCKET serverSocket,
            char args[MAX_ARGS][DEFAULT_BUFLEN],
            int argsCount )
```

Sends command to server then displays the response if no error. Here is the caller graph for this function:

2.4 misc.c File Reference

`#include "misc.h"`
Include dependency graph for misc.c:



Functions

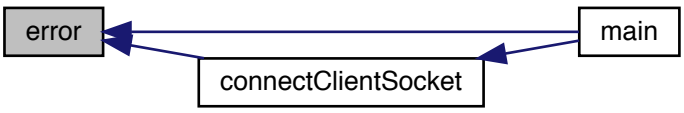
- void error (const char *msg)
- void ZombieKill (int sig)

2.4.1 Function Documentation

2.4.1.1 error()

```
void error (
            const char * msg )
```

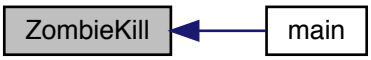
Prints error then closes the program. Here is the caller graph for this function:



2.4.1.2 ZombieKill()

```
void ZombieKill (
            int sig )
```

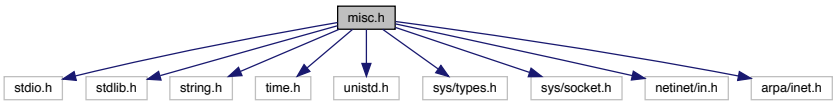
If any zombie signals it is then killed asap. Here is the caller graph for this function:



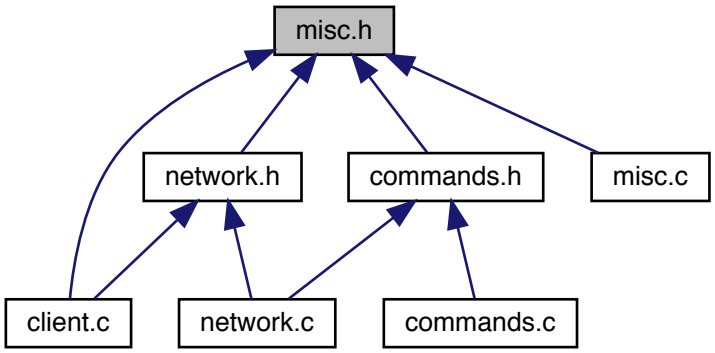
2.5 misc.h File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
```

Include dependency graph for misc.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define DEFAULT_BUFLEN 1024
- #define MAX_ARGS 10
- #define V_SOCKET int
- #define INVALID_SOCKET -1

Functions

- void error (const char *msg)
- void ZombieKill (int sig)

2.5.1 Macro Definition Documentation

2.5.1.1 DEFAULT_BUFLEN

```
#define DEFAULT_BUFLEN 1024
```

2.5.1.2 INVALID_SOCKET

```
#define INVALID_SOCKET -1
```

2.5.1.3 MAX_ARGS

```
#define MAX_ARGS 10
```

2.5.1.4 V_SOCKET

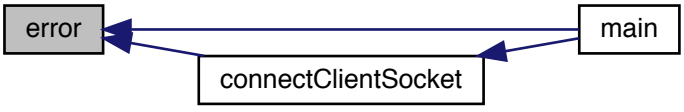
```
#define V_SOCKET int
```

2.5.2 Function Documentation

2.5.2.1 error()

```
void error (
            const char * msg )
```

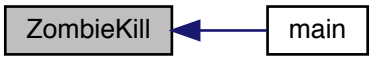
Prints error then closes the program. Here is the caller graph for this function:



2.5.2.2 ZombieKill()

```
void ZombieKill (
            int sig )
```

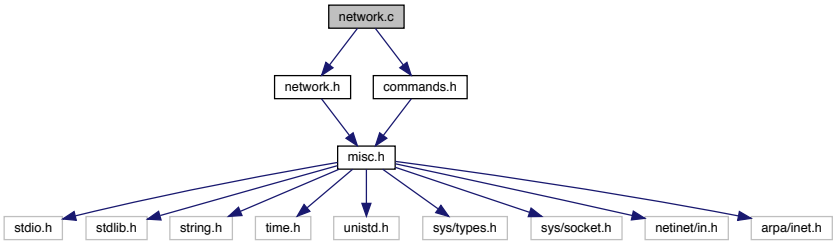
If any zombie signals it is then killed asap. Here is the caller graph for this function:



2.6 network.c File Reference

```
#include "network.h"
#include "commands.h"
```

Include dependency graph for network.c:



Functions

- V_SOCKET connectClientSocket (char *hostname, int portNum)
- int manageCommand (V_SOCKET serverSocket, char buffer[DEFAULT_BUFLEN])

2.6.1 Function Documentation

2.6.1.1 connectClientSocket()

```
V_SOCKET connectClientSocket (
            char * hostname,
            int portNum )
```

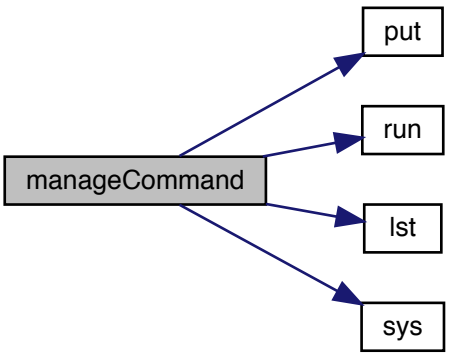
Connects to the server and returns socket descriptor. Here is the call graph for this function:

Here is the caller graph for this function:

2.6.1.2 manageCommand()

```
int manageCommand (
            V_SOCKET serverSocket,
            char buffer[DEFAULT_BUFLEN] )
```

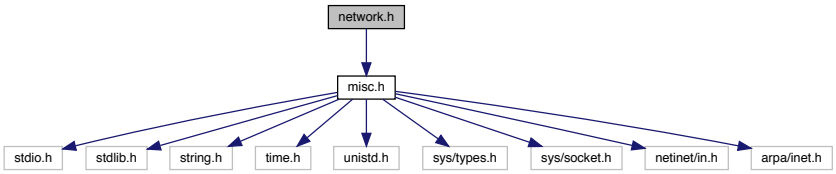
Turns raw string into command and args then calls appropriate function. Here is the call graph for this function:



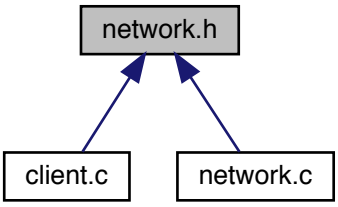
Here is the caller graph for this function:

2.7 network.h File Reference

`#include "misc.h"`
Include dependency graph for network.h:



This graph shows which files directly or indirectly include this file:



Functions

- V_SOCKET connectClientSocket (char ∗hostname, int portNum)
- int manageCommand (V_SOCKET serverSocket, char buffer[DEFAULT_BUFLEN])

2.7.1 Function Documentation

2.7.1.1 connectClientSocket()

```
V_SOCKET connectClientSocket (
            char * hostname,
            int portNum )
```

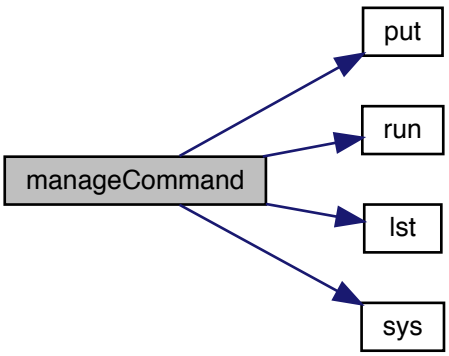
Connects to the server and returns socket descriptor. Here is the call graph for this function:

Here is the caller graph for this function:

2.7.1.2 manageCommand()

```
int manageCommand (
            V_SOCKET serverSocket,
            char buffer[DEFAULT_BUFLEN] )
```

Turns raw string into command and args then calls appropriate function. Here is the call graph for this function:



Here is the caller graph for this function:

2.8 winchild.c File Reference

Index