

Server

1

Generated by Doxygen 1.8.14

Contents

| | | |
|----------|-------------------------------------|----------|
| 1 | File Index | 1 |
| 1.1 | File List | 1 |
| 2 | File Documentation | 3 |
| 2.1 | commands.c File Reference | 3 |
| 2.1.1 | Function Documentation | 3 |
| 2.1.1.1 | checkProject() | 4 |
| 2.1.1.2 | get() | 4 |
| 2.1.1.3 | getList() | 4 |
| 2.1.1.4 | getSourceFiles() | 5 |
| 2.1.1.5 | lst() | 5 |
| 2.1.1.6 | put() | 6 |
| 2.1.1.7 | run() | 6 |
| 2.1.1.8 | sys() | 7 |
| 2.2 | commands.h File Reference | 7 |
| 2.2.1 | Function Documentation | 8 |
| 2.2.1.1 | checkProject() | 8 |
| 2.2.1.2 | get() | 8 |
| 2.2.1.3 | getList() | 9 |
| 2.2.1.4 | getSourceFiles() | 9 |
| 2.2.1.5 | lst() | 9 |
| 2.2.1.6 | put() | 10 |
| 2.2.1.7 | run() | 11 |
| 2.2.1.8 | sys() | 11 |

| | | |
|---------|--------------------------------|----|
| 2.3 | misc.c File Reference | 12 |
| 2.3.1 | Function Documentation | 12 |
| 2.3.1.1 | error() | 12 |
| 2.3.1.2 | ZombieKill() | 13 |
| 2.4 | misc.h File Reference | 13 |
| 2.4.1 | Macro Definition Documentation | 14 |
| 2.4.1.1 | DEFAULT_BUFLLEN | 14 |
| 2.4.1.2 | INVALID_SOCKET | 14 |
| 2.4.1.3 | MAX_ARGS | 15 |
| 2.4.1.4 | V_SOCKET | 15 |
| 2.4.2 | Function Documentation | 15 |
| 2.4.2.1 | error() | 15 |
| 2.4.2.2 | ZombieKill() | 15 |
| 2.5 | network.c File Reference | 16 |
| 2.5.1 | Function Documentation | 16 |
| 2.5.1.1 | acceptNewConnection() | 16 |
| 2.5.1.2 | createServerSocket() | 17 |
| 2.5.1.3 | manageCommand() | 17 |
| 2.5.1.4 | manageConnection() | 18 |
| 2.6 | network.h File Reference | 19 |
| 2.6.1 | Function Documentation | 19 |
| 2.6.1.1 | acceptNewConnection() | 19 |
| 2.6.1.2 | createServerSocket() | 20 |
| 2.6.1.3 | manageCommand() | 21 |
| 2.6.1.4 | manageConnection() | 21 |
| 2.7 | server.c File Reference | 22 |
| 2.7.1 | Macro Definition Documentation | 23 |
| 2.7.1.1 | PORT | 23 |
| 2.7.2 | Function Documentation | 23 |
| 2.7.2.1 | main() | 23 |
| 2.8 | winchild.c File Reference | 23 |

Chapter 1

File Index

1.1 File List

Here is a list of all files with brief descriptions:

| | |
|----------------------------|----|
| commands.c | 3 |
| commands.h | 7 |
| misc.c | 12 |
| misc.h | 13 |
| network.c | 16 |
| network.h | 19 |
| server.c | 22 |
| winchild.c | 23 |

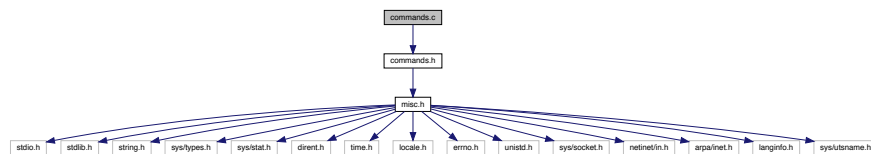
Chapter 2

File Documentation

2.1 commands.c File Reference

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

Include dependency graph for commands.c:



Functions

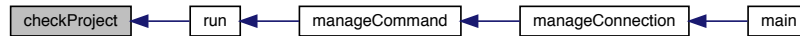
- int [checkProject](#) (char directory[[DEFAULT_BUFLen](#)])
- void [getSourceFiles](#) (char directory[[DEFAULT_BUFLen](#)], char cfiles[[MAX_ARGS](#)][[DEFAULT_BUFLen](#)], int *cNum)
- void [getList](#) (char directory[[DEFAULT_BUFLen](#)], char files[][[DEFAULT_BUFLen](#)], int *num, int longlist, int programlist)
- int [put](#) ([V_SOCKET](#) clientSocket, char args[[MAX_ARGS](#)][[DEFAULT_BUFLen](#)], int argsCount)
- int [get](#) ([V_SOCKET](#) clientSocket, char args[[MAX_ARGS](#)][[DEFAULT_BUFLen](#)], int argsCount)
- int [run](#) ([V_SOCKET](#) clientSocket, char args[[MAX_ARGS](#)][[DEFAULT_BUFLen](#)], int argsCount)
- int [lst](#) ([V_SOCKET](#) clientSocket, char args[[MAX_ARGS](#)][[DEFAULT_BUFLen](#)], int argsCount)
- int [sys](#) ([V_SOCKET](#) clientSocket, char args[[MAX_ARGS](#)][[DEFAULT_BUFLen](#)], int argsCount)

2.1.1 Function Documentation

2.1.1.1 checkProject()

```
int checkProject (
    char directory[DEFAULT_BUFLen] )
```

Scans through the given directory to determine if the program need recompiling. It does this by checking dates on .c files and executables. Returns 1 if needs compiling 0 otherwise. Here is the caller graph for this function:



2.1.1.2 get()

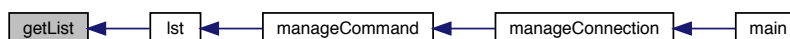
```
int get (
    V_SOCKET clientSocket,
    char args[MAX_ARGS][DEFAULT_BUFLen],
    int argsCount )
```

Validates command then returns contents of file requested.

2.1.1.3 getList()

```
void getList (
    char directory[DEFAULT_BUFLen],
    char files[][DEFAULT_BUFLen],
    int * num,
    int longlist,
    int programlist )
```

Gets a list of files from give directory. If programlist == 1 will print files inside otherwise just directoies. If longlist == 1 then more details will be stored. Here is the caller graph for this function:



2.1.1.4 getSourceFiles()

```
void getSourceFiles (
    char directory[DEFAULT_BUFLen],
    char cfiles[MAX_ARGS][DEFAULT_BUFLen],
    int * cNum )
```

Scans a directory and stores file paths and number of files in the given pointers. Here is the caller graph for this function:



2.1.1.5 lst()

```
int lst (
    V_SOCKET clientSocket,
    char args[MAX_ARGS][DEFAULT_BUFLen],
    int argsCount )
```

Validates command then sends a list of files or program. if -l will send more details. Here is the call graph for this function:



Here is the caller graph for this function:



2.1.1.6 put()

```
int put (
    V_SOCKET clientSocket,
    char args[MAX_ARGS][DEFAULT_BUFLLEN],
    int argsCount )
```

Validates put command args are correct.

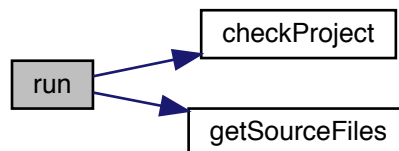
Then checks if -f is specified to see if overwriting is okay. It then sends a message to the client saying it ready for files. it then reads in all files and stores them under program directory. Here is the caller graph for this function:



2.1.1.7 run()

```
int run (
    V_SOCKET clientSocket,
    char args[MAX_ARGS][DEFAULT_BUFLLEN],
    int argsCount )
```

Determines if file needs to be recompiled. Then compiles if needed if any errors it returns the contents of the error message. Here is the call graph for this function:



Here is the caller graph for this function:



2.1.1.8 sys()

```
int sys (
    V_SOCKET clientSocket,
    char args[MAX_ARGS][DEFAULT_BUFLen],
    int argsCount )
```

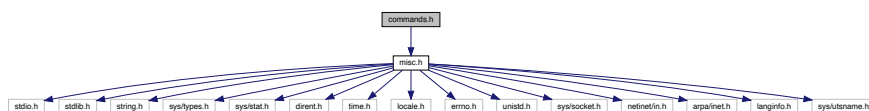
Sends details of the server host. Here is the caller graph for this function:



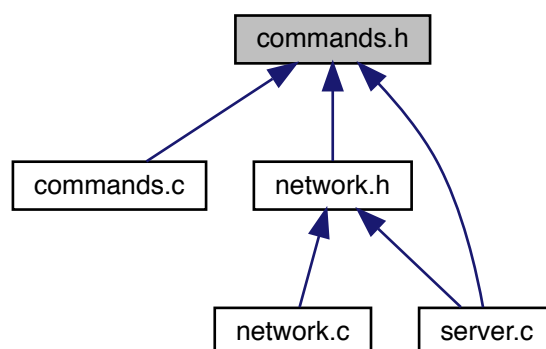
2.2 commands.h File Reference

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

Include dependency graph for commands.h:



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



Functions

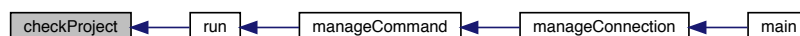
- int [checkProject](#) (char directory[[DEFAULT_BUFLen](#)])
- void [getSourceFiles](#) (char directory[[DEFAULT_BUFLen](#)], char cfiles[[MAX_ARGS](#)][[DEFAULT_BUFLen](#)], int *cNum)
- void [getList](#) (char directory[[DEFAULT_BUFLen](#)], char files[][[DEFAULT_BUFLen](#)], int *num, int longlist, int programlist)
- int [put](#) ([V_SOCKET](#) clientSocket, char args[[MAX_ARGS](#)][[DEFAULT_BUFLen](#)], int argsCount)
- int [get](#) ([V_SOCKET](#) clientSocket, char args[[MAX_ARGS](#)][[DEFAULT_BUFLen](#)], int argsCount)
- int [run](#) ([V_SOCKET](#) clientSocket, char args[[MAX_ARGS](#)][[DEFAULT_BUFLen](#)], int argsCount)
- int [lst](#) ([V_SOCKET](#) clientSocket, char args[[MAX_ARGS](#)][[DEFAULT_BUFLen](#)], int argsCount)
- int [sys](#) ([V_SOCKET](#) clientSocket, char args[[MAX_ARGS](#)][[DEFAULT_BUFLen](#)], int argsCount)

2.2.1 Function Documentation

2.2.1.1 [checkProject\(\)](#)

```
int checkProject (
    char directory[DEFAULT\_BUFLen] )
```

Scans through the given directory to determine if the program need recompiling. It does this by checking dates on .c files and executables. Returns 1 if needs compiling 0 otherwise. Here is the caller graph for this function:



2.2.1.2 [get\(\)](#)

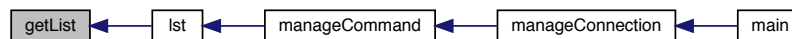
```
int get (
    V\_SOCKET clientSocket,
    char args[MAX\_ARGS][DEFAULT\_BUFLen],
    int argsCount )
```

Validates command then returns contents of file requested.

2.2.1.3 getList()

```
void getList (
    char directory[DEFAULT_BUFLen],
    char files[][DEFAULT_BUFLen],
    int * num,
    int longlist,
    int programlist )
```

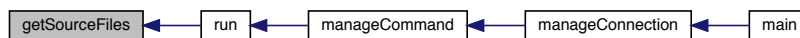
Gets a list of files from give directory. If programlist == 1 will print files inside otherwise just directoies. If longlist == 1 then more details will be stored. Here is the caller graph for this function:



2.2.1.4 getSourceFiles()

```
void getSourceFiles (
    char directory[DEFAULT_BUFLen],
    char cfiles[MAX_ARGS][DEFAULT_BUFLen],
    int * cNum )
```

Scans a directory and stores file paths and number of files in the given pointers. Here is the caller graph for this function:



2.2.1.5 lst()

```
int lst (
    V_SOCKET clientSocket,
    char args[MAX_ARGS][DEFAULT_BUFLen],
    int argsCount )
```

Validates command then sends a list of files or program. if -l will send more details. Here is the call graph for this function:



Here is the caller graph for this function:



2.2.1.6 put()

```
int put (
    V_SOCKET clientSocket,
    char args[MAX_ARGS][DEFAULT_BUFLen],
    int argsCount )
```

Validates put command args are correct.

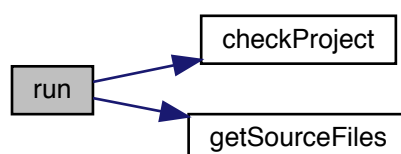
Then checks if -f is specified to see if overwriting is okay. It then sends a message to the client saying it ready for files. it then reads in all files and stores them under program directory. Here is the caller graph for this function:



2.2.1.7 run()

```
int run (  
    V_SOCKET clientSocket,  
    char args[MAX_ARGS][DEFAULT_BUFLLEN],  
    int argsCount )
```

Determines if file needs to be recompiled. Then compiles if needed if any errors it returns the contents of the error message. Here is the call graph for this function:



Here is the caller graph for this function:



2.2.1.8 sys()

```
int sys (  
    V_SOCKET clientSocket,  
    char args[MAX_ARGS][DEFAULT_BUFLLEN],  
    int argsCount )
```

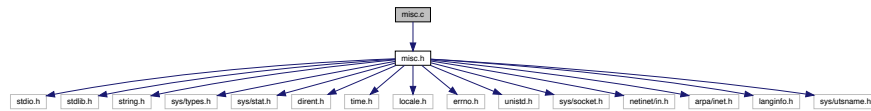
Sends details of the server host. Here is the caller graph for this function:



2.3 misc.c File Reference

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

Include dependency graph for misc.c:



Functions

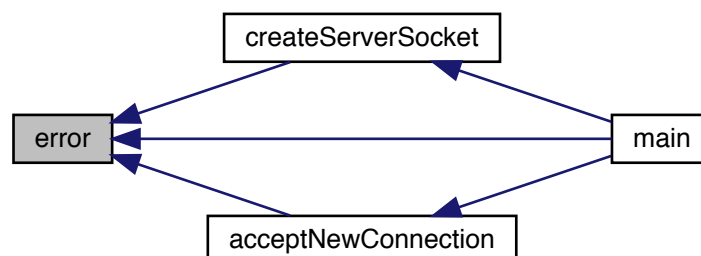
- void [error](#) (const char *msg)
- void [ZombieKill](#) (int sig)

2.3.1 Function Documentation

2.3.1.1 error()

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

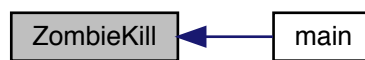
Prints error then closes the program. Here is the caller graph for this function:



2.3.1.2 ZombieKill()

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

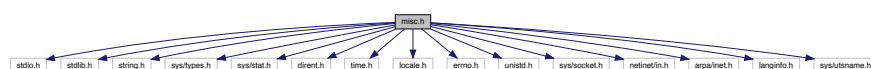
If any zombie signals it is then killed asap. Here is the caller graph for this function:



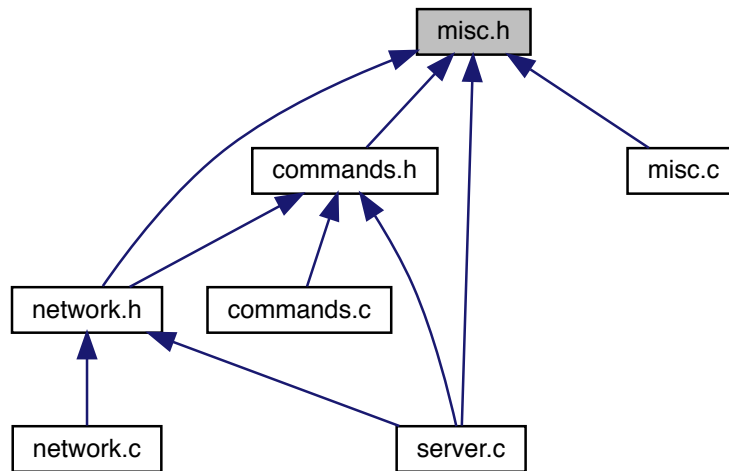
2.4 misc.h File Reference

```
#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#include <sys/types.h>  
#include <sys/stat.h>  
#include <dirent.h>  
#include <time.h>  
#include <locale.h>  
#include <errno.h>  
#include <unistd.h>  
#include <sys/socket.h>  
#include <netinet/in.h>  
#include <arpa/inet.h>  
#include <langinfo.h>  
#include <sys/utsname.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)

2.4.1 Macro Definition Documentation

2.4.1.1 DEFAULT_BUFLen

```
#define DEFAULT_BUFLen 1024
```

2.4.1.2 INVALID_SOCKET

```
#define INVALID_SOCKET -1
```

2.4.1.3 MAX_ARGS

```
#define MAX_ARGS 10
```

2.4.1.4 V_SOCKET

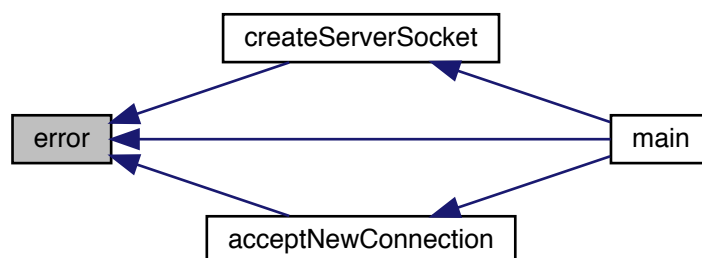
```
#define V_SOCKET int
```

2.4.2 Function Documentation

2.4.2.1 error()

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

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



2.4.2.2 ZombieKill()

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

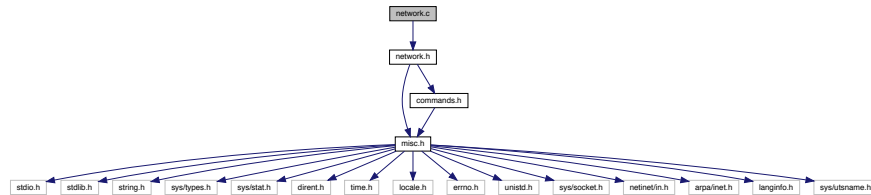
If any zombie signals it is then killed asap. Here is the caller graph for this function:



2.5 network.c File Reference

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

Include dependency graph for network.c:



Functions

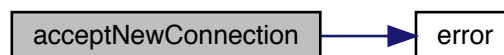
- [V_SOCKET createServerSocket](#) (int portNum)
- [V_SOCKET acceptNewConnection](#) ([V_SOCKET](#) sockfd)
- void [manageConnection](#) ([V_SOCKET](#) clientSocket)
- int [manageCommand](#) ([V_SOCKET](#) clientSocket, char buffer[[DEFAULT_BUFLen](#)])

2.5.1 Function Documentation

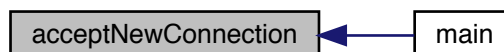
2.5.1.1 acceptNewConnection()

```
V_SOCKET acceptNewConnection (
    V_SOCKET sockfd )
```

Accepts a connection. For both windows and unix. Here is the call graph for this function:



Here is the caller graph for this function:



2.5.1.2 createServerSocket()

```
V_SOCKET createServerSocket (  
    int portNum )
```

Creates a socket. For both windows and unix. Here is the call graph for this function:



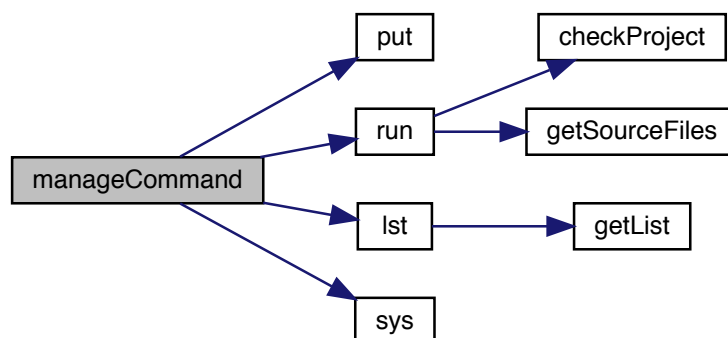
Here is the caller graph for this function:



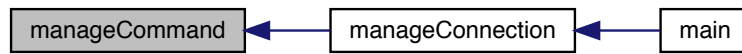
2.5.1.3 manageCommand()

```
int manageCommand (  
    V_SOCKET clientSocket,  
    char buffer[DEFAULT_BUFLLEN] )
```

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:

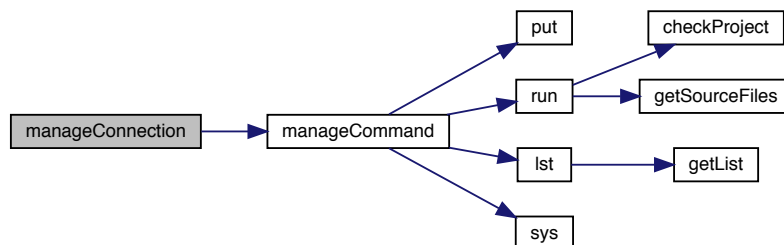


2.5.1.4 manageConnection()

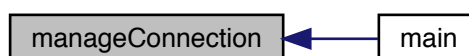
```

void manageConnection (
    V_SOCKET clientSocket )
  
```

Receives first command from client then calls manageCommand with the message. Here is the call graph for this function:

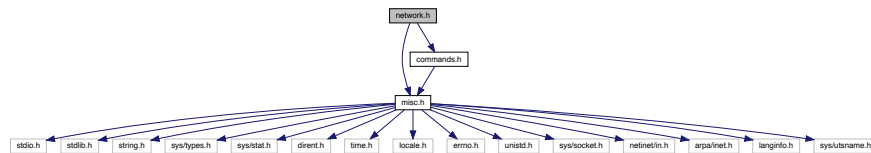


Here is the caller graph for this function:

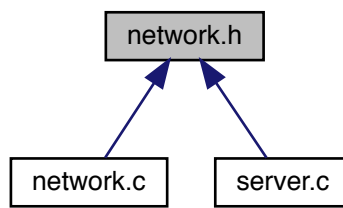


2.6 network.h File Reference

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



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



Functions

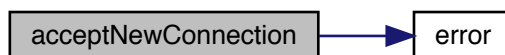
- [V_SOCKET createServerSocket](#) (int portNum)
- [V_SOCKET acceptNewConnection](#) ([V_SOCKET](#) sockfd)
- void [manageConnection](#) ([V_SOCKET](#) clientSocket)
- int [manageCommand](#) ([V_SOCKET](#) clientSocket, char buffer[[DEFAULT_BUFLN](#)])

2.6.1 Function Documentation

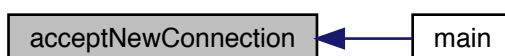
2.6.1.1 acceptNewConnection()

```
V\_SOCKET acceptNewConnection (
    V\_SOCKET sockfd )
```

Accepts a connection. For both windows and unix. Here is the call graph for this function:



Here is the caller graph for this function:



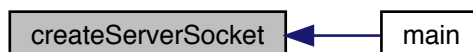
2.6.1.2 `createServerSocket()`

```
V_SOCKET createServerSocket (  
    int portNum )
```

Creates a socket. For both windows and unix. Here is the call graph for this function:



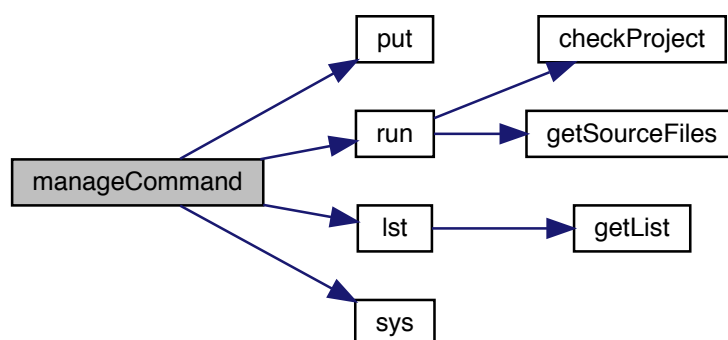
Here is the caller graph for this function:



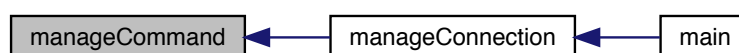
2.6.1.3 manageCommand()

```
int manageCommand (
    V_SOCKET clientSocket,
    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:

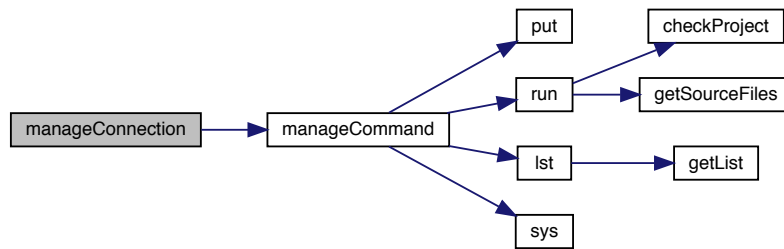


2.6.1.4 manageConnection()

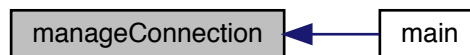
```
void manageConnection (
    V_SOCKET clientSocket )
```

Receives first command from client then calls `manageCommand` with the message. Here is the call graph for this

function:

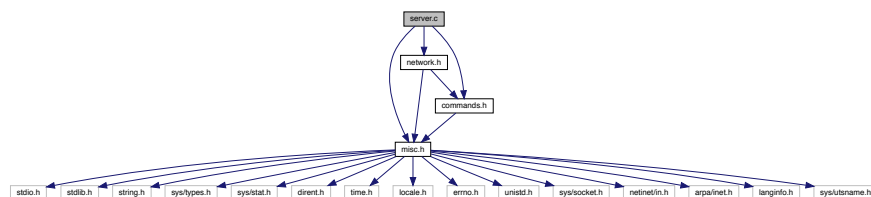


Here is the caller graph for this function:



2.7 server.c File Reference

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



Macros

- `#define` `PORT` 80

Functions

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

2.7.1 Macro Definition Documentation

2.7.1.1 PORT

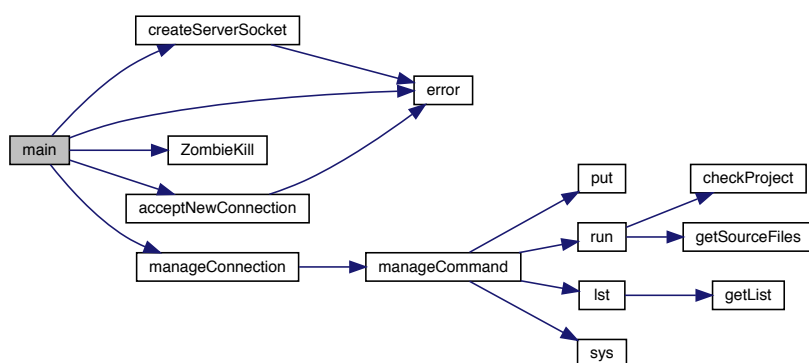
```
#define PORT 80
```

2.7.2 Function Documentation

2.7.2.1 main()

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

Loops accepting connection then forks and runs manageCommand. Here is the call graph for this function:



2.8 winchild.c File Reference

Index

acceptNewConnection
 network.c, 16
 network.h, 19

checkProject
 commands.c, 3
 commands.h, 8

commands.c, 3
 checkProject, 3
 get, 4
 getList, 4
 getSourceFiles, 4
 lst, 5
 put, 5
 run, 6
 sys, 6

commands.h, 7
 checkProject, 8
 get, 8
 getList, 8
 getSourceFiles, 9
 lst, 9
 put, 10
 run, 10
 sys, 11

createServerSocket
 network.c, 16
 network.h, 20

DEFAULT_BUFLen
 misc.h, 14

error
 misc.c, 12
 misc.h, 15

get
 commands.c, 4
 commands.h, 8

getList
 commands.c, 4
 commands.h, 8

getSourceFiles
 commands.c, 4
 commands.h, 9

INVALID_SOCKET
 misc.h, 14

lst
 commands.c, 5

commands.h, 9

MAX_ARGS
 misc.h, 14

main
 server.c, 23

manageCommand
 network.c, 17
 network.h, 20

manageConnection
 network.c, 18
 network.h, 21

misc.c, 12
 error, 12
 ZombieKill, 12

misc.h, 13
 DEFAULT_BUFLen, 14
 error, 15
 INVALID_SOCKET, 14
 MAX_ARGS, 14
 V_SOCKET, 15
 ZombieKill, 15

network.c, 16
 acceptNewConnection, 16
 createServerSocket, 16
 manageCommand, 17
 manageConnection, 18
network.h, 19
 acceptNewConnection, 19
 createServerSocket, 20
 manageCommand, 20
 manageConnection, 21

PORT
 server.c, 23

put
 commands.c, 5
 commands.h, 10

run
 commands.c, 6
 commands.h, 10

server.c, 22
 main, 23
 PORT, 23

sys
 commands.c, 6
 commands.h, 11

V_SOCKET
misc.h, [15](#)

winchild.c, [23](#)

ZombieKill
misc.c, [12](#)
misc.h, [15](#)