

CSC415 – Windows/Linux Shell

Aleksandr Kibis

9/30/2014

This project was the hardest one assigned to us yet, but very useful since I learned quite a lot of different functions. As with the previous projects, the Linux version was much easier to implement than the Windows counterpart. However, this time around, the Windows version was actually less complex. In the Linux version, an infinite loop was created that created a child process and waited for it to finish executing before spawning another. Various if-statements were used in order to check for process id and whether the user issued the “exit” command or not. The biggest difference that I noticed is that the users input had to be tokenized manually for POSIX, in Win32, the CreateProcess functions took care of this automatically. This difference also led to a need for a two-dimensional array in POSIX.

While the Windows version was eventually less complicated, it took me a lot longer to figure out exactly how to pass the correct arguments to CreateProcess. This is something that really bothers me about the Win32 API. There are just so many different variable types and most are not capable of being cast into anything simple. For example, in order to properly pass arguments to CreateProcess, I had to send it a TCHAR array instead of a char array. Char* cannot be cast to TCHAR, instead the MultiByteToWideChar function has to be used. It’s nice that CreateProcess takes care of everything but chasing types in Win32 is like going down a rabbit hole.

POSIX

/*

* File: linux_shell.c

* Author: netdom

*

* Created on September 28, 2014, 7:40 PM

*/

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

```

#define EXIT_SUCCESS 0

#define EXIT_FAILURE 1

#define BUFFER_SIZE 2048

#define MAX_ARGS 20

/*
 *
 */

int main() {

    char input[BUFFER_SIZE]; // single string of user args

    const char s[2] = "\n "; // string delimiter

    char* token; // single token from string

    int myargc = 0; // number of args

    char** myargv;

    pid_t cpid = 0;

    int status = 0, error;

    while (1) {

        // reset argument memory

        myargv = (char**)calloc(MAX_ARGS, sizeof(char*));

        myargc = 0;

        // start shell gui and get user input

        printf("netdom> ");

        fgets(input, BUFFER_SIZE, stdin);

        token = strtok(input, s); // tokenize user input

        // fork process for each new input

        while (token != NULL) {

            myargv[myargc++] = token;

            token = strtok(NULL, s);

            if (strcmp(myargv[0], "exit") == 0) { // exit shell upon keyword

```

```

    printf("<3\n");

    exit(EXIT_SUCCESS);
}

if ((cpid = fork()) < 0) {

    perror("Error sporking process o_O\n");

    exit(EXIT_FAILURE);

}

else if (cpid == 0) { // limit to only 1 shell process

    if (error = execvp(*myargv, myargv) < 0) {

        perror("EXEC ERROR\n");

        exit(EXIT_FAILURE);

    }

} else {

    while (wait(&status) != cpid); // have parent wait for child to finish

}

//printf("myargv[%d]: %s\n", myargc - 1, myargv[myargc - 1]);

}

free(myargv); // free memory

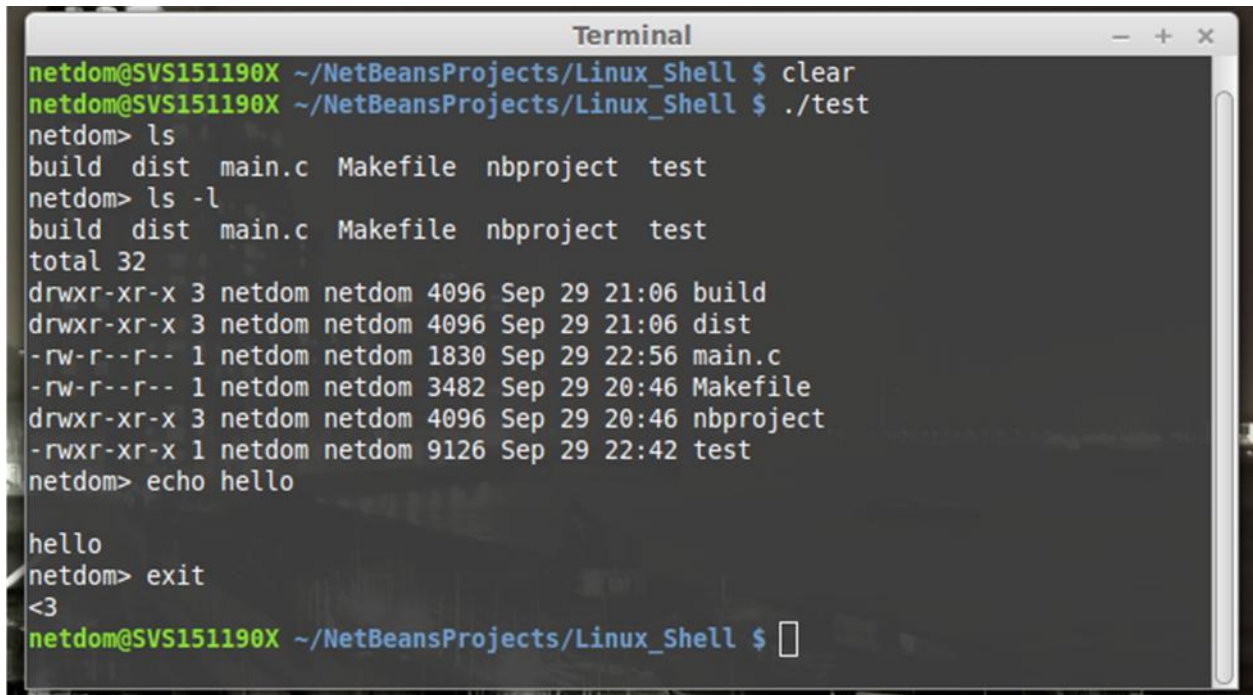
}

return (EXIT_SUCCESS);

}

```

Output

A terminal window titled "Terminal" with standard window controls. The prompt is "netdom@SVS151190X ~/NetBeansProjects/Linux_Shell \$". The user enters "clear", then "./test". The prompt changes to "netdom>". The user enters "ls", showing a directory listing of files: build, dist, main.c, Makefile, nbproject, and test. Then the user enters "ls -l", showing a detailed directory listing with permissions, owner, group, size, date, and filename. Finally, the user enters "echo hello", which outputs "hello", and then "exit", which outputs "<3". The prompt returns to "netdom@SVS151190X ~/NetBeansProjects/Linux_Shell \$".

```
netdom@SVS151190X ~/NetBeansProjects/Linux_Shell $ clear
netdom@SVS151190X ~/NetBeansProjects/Linux_Shell $ ./test
netdom> ls
build dist main.c Makefile nbproject test
netdom> ls -l
build dist main.c Makefile nbproject test
total 32
drwxr-xr-x 3 netdom netdom 4096 Sep 29 21:06 build
drwxr-xr-x 3 netdom netdom 4096 Sep 29 21:06 dist
-rw-r--r-- 1 netdom netdom 1830 Sep 29 22:56 main.c
-rw-r--r-- 1 netdom netdom 3482 Sep 29 20:46 Makefile
drwxr-xr-x 3 netdom netdom 4096 Sep 29 20:46 nbproject
-rwxr-xr-x 1 netdom netdom 9126 Sep 29 22:42 test
netdom> echo hello

hello
netdom> exit
<3
netdom@SVS151190X ~/NetBeansProjects/Linux_Shell $
```

Win32

/*

File: windows_shell.c

Author: Aleksandr Kibis

Date: 9/29/2014

This program mimicks the windows command prompt

*/

```
#include <Windows.h>
```

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
#include <tchar.h>
```

```
#define EXIT_SUCCESS 0
```

```
#define EXIT_FAILURE 1
```

```
#define BUFFER_SIZE 2048
```

```

int main()

{

    char* input;

    TCHAR command[BUFFER_SIZE];

    STARTUPINFO si;

    PROCESS_INFORMATION pi;

    ZeroMemory( &si, sizeof(si) );

    si.cb = sizeof(si);

    ZeroMemory( &pi, sizeof(pi) );

    while(1){

        // allocate memory and add cmd preprocessor

        char args[BUFFER_SIZE + 4] = "/c ";

        input = (char*)calloc(BUFFER_SIZE, sizeof(char));

        printf("netdom> ");

        // get user input and concatenate with preprocessor

        fgets(input, BUFFER_SIZE, stdin);

        if (strcmp(input, "exit\n") == 0) { // exit shell upon keyword

            printf("<3\n");

            exit(EXIT_SUCCESS);

        }

        strcat(args, input);

        //printf("stringcat: %s\n",add);

        // convert args from char* to TCHAR*

        MultiByteToWideChar(CP_UTF8, 0, args, -1, command, BUFFER_SIZE + 4);

        if( !CreateProcess(TEXT("C:\\Windows\\System32\\cmd.exe"), // Use Windows Command Prompt

            command, // Pass command line args

            NULL, // Process handle not inheritable

            NULL, // Thread handle not inheritable

            FALSE, // Set handle inheritance to FALSE

```

```

        0,        // default flags - Normal Priority

        NULL,     // Use parent's environment block

        NULL,     // Use parent's starting directory

        &si,      // Pointer to STARTUPINFO structure

        &pi )     // Pointer to PROCESS_INFORMATION structure

    )

{

    printf( "CreateProcess failed (%d).\n", GetLastError() );

    exit(EXIT_FAILURE);

}

//printf("check\n");

// Wait until child process exits.

WaitForSingleObject( pi.hProcess, INFINITE );

// Close process and thread handles.

CloseHandle( pi.hProcess );
CloseHandle( pi.hThread );

//printf("you typed: %s\n", input);

free(input);

}

return 0;

}

```

100

```
C:\WINDOWS\system32\cmd.exe
netdom> dir
Volume in drive C has no label.
Volume Serial Number is 4E3E-3157

Directory of C:\Users\rusky\Dropbox\Fall 2014\415\Windows Shell\Windows Shell

09/29/2014  11:46 PM    <DIR>          .
09/29/2014  11:46 PM    <DIR>          ..
09/30/2014  11:32 AM    <DIR>          Debug
09/30/2014  11:32 AM                2,053 main.c
09/30/2014  01:56 AM                4,100 Windows Shell.vcxproj
09/29/2014  11:46 PM                951 Windows Shell.vcxproj.filters
               3 File(s)                7,104 bytes
               3 Dir(s)  145,648,439,296 bytes free

netdom> exit
<3
Press any key to continue . . . _
```