Artem Tolstov

March 11, 2015

Homework 3

CSC 415

Win32 code

```
#include <stdio.h>
#include <stdlib.h>
#include <windows.h>
#include <string.h>
#define BUFFER_SIZE 1024

int main()
{

    char buffer[BUFFER_SIZE];
    STARTUPINFO si;
    PROCESS_INFORMATION pi;
    char* commandLine = NULL;
    char* nextToken = NULL;

    /* allocate memory */
    ZeroMemory(&si, sizeof(si));
    si.cb = sizeof(si);
    ZeroMemory(&pi, sizeof(pi));



    printf("Loop begins\n");
    while(1){
        //prints shell prompt, takes input
        printf("Myshell> ");
             fgets(buffer, BUFFER_SIZE, stdin);
```

```
    //scanf("%s", buffer);
    //printf("\nbuffer contains: %s \n\n", buffer);

    //used to tokenize our command up to when we hit enter
    commandLine = strtok_s(buffer, "\n", &nextToken);

    //exits the while loop
    if(strcmp(buffer, "exit") == 0)
    break;

          //creates a process that will send command to outer shell
          CreateProcess
          (
             NULL,
             commandLine,
             NULL,
             NULL,
             FALSE,
             0,
             NULL,
             NULL,
             &si,
             &pi
    );

          // waits for created process to finish
          WaitForSingleObject(pi.hProcess, INFINITE);
          //printf("**Child Complete**\n");


}
//printf("END WHILE LOOP\n");
```

```
        CloseHandle(pi.hProcess);
    CloseHandle(pi.hThread);
    return 0;
}
```

Description of win32 code:

A character array of BUFFER_SIZE is created, where buffer size is the predetermined 1024 byte specified in the instructions. After allocating memory resources and initializing the required variables the program enters an infinite loop. The loop outputs the myshell> and awaits input from the user. Fgets() is used to capture the user input and is then placed into a token called commandLine which is a pointer to a character. The string in buffer is compared to exit so the loop can be exited, which is done by a simple break to leave the while loop. Finally CreateProcess is called, it creates a child process which sends the contents of CommandLine token to the outer shell and prints the response (as the child) while the parent waits infinitely until the child completes it's task. Normally the while loop would repeat again unless it was exited.

Console Output Win32

```
c:\Users\czar__000\Desktop\CSC 415 Operating Systems\csc415hw3>cl hw3win32.c
Microsoft (R) C/C++ Optimizing Compiler Version 17.00.61030 for x86
Copyright (C) Microsoft Corporation.  All rights reserved.

hw3win32.c
Microsoft (R) Incremental Linker Version 11.00.61030.0
Copyright (C) Microsoft Corporation.  All rights reserved.

/out:hw3win32.exe
hw3win32.obj

c:\Users\czar__000\Desktop\CSC 415 Operating Systems\csc415hw3>hw3win32
Loop begins
Myshell> ls
bin            csc415hw3.depend  hw3win32.c    hw3win32.obj  main.exe  obj
csc415hw3.cbp  csc415hw3.layout  hw3win32.exe  main.c        main.obj
```

```
Myshell> ls -l
total 150
drwxrwx---+ 1 czar__000 None     0 Mar 10 19:49 bin
-rwxrwx---+ 1 czar__000 None  1116 Mar 11 04:32 csc415hw3.cbp
-rwxrwx---+ 1 czar__000 None   148 Mar 11 00:26 csc415hw3.depend
-rwxrwx---+ 1 czar__000 None   321 Mar 11 19:29 csc415hw3.layout
-rwxrwx---+ 1 czar__000 None  1441 Mar 11 22:40 hw3win32.c
-rwxrwx---+ 1 czar__000 None 61952 Mar 11 23:50 hw3win32.exe
-rwxrwx---+ 1 czar__000 None  1585 Mar 11 23:50 hw3win32.obj
-rwxrwx---+ 1 czar__000 None  1441 Mar 11 22:40 main.c
-rwxrwx---+ 1 czar__000 None 61952 Mar 11 22:41 main.exe
-rwxrwx---+ 1 czar__000 None  1581 Mar 11 22:41 main.obj
drwxrwx---+ 1 czar__000 None     0 Mar 10 19:49 obj
Myshell> comp
Name of first file to compare: hw3win32.c
Name of second file to compare: hw3win32.exe
Option:
Comparing hw3win32.c and hw3win32.exe...
Files are different sizes.

Compare more files (Y/N) ? n
Myshell> exit

c:\Users\czar__000\Desktop\CSC 415 Operating Systems\csc415hw3>
```

Posix Code

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#define BUFFER_SIZE 1024

int main()
```

```
{
 pid_t pid = 0;
 char buffer[BUFFER_SIZE];
 char* token = NULL;
 char* args[5];
 int counter = 0;



 while(1)
 {
   printf("myshell>");
   //scanf("%s", buffer);
   fgets(buffer, BUFFER_SIZE, stdin);
   //printf("buffer contains: %s\n", buffer);
   //if(strcmp(buffer, "exit") == 0)
   //  exit(0);


   token = (char *)strtok(buffer, " \n");
   counter = 0;
   while(token != NULL)
   {
     args[counter] = token;
     token = (char *)strtok(NULL, " \n");
     counter++;
   }
   args[counter] = NULL;
   if(strcmp(args[0], "exit") == 0)
       exit(0);


   pid = fork();
   if (pid == 0)
   {
```

```
    //printf("Child process says HI\n");
    execlp(args[0], args[0], args[1], args[2], args[3], (char *)NULL);
    exit(-1);
   }
   else
   {
    wait(NULL);
   }

 }
printf("End While Loop\n\n");
return 0;
}
```

Explanation of Posix code:

For the most part the Posix version of the code is the same as the win32 version with the exception of how the user input is tokenized and that the posix version uses fork instead of create process. After fork creates it's child execlp() is called to send our arguments to the outer shell.

Console Output

```
liteshotv3@liteshotv3-VirtualBox:~/Desktop/CSC415/HW3$ make hw3posix
cc     hw3posix.c   -o hw3posix
liteshotv3@liteshotv3-VirtualBox:~/Desktop/CSC415/HW3$ ./hw3posix
myshell>ls
hw3.c~    hw3posix  hw3posix.c  hw3posix.c~
myshell>ls -l
total 20
-rw-rw-r-- 1 liteshotv3 liteshotv3  903 Mar 11 23:25 hw3.c~
-rwxrwxr-x 1 liteshotv3 liteshotv3 7589 Mar 11 23:34 hw3posix
-rw-rw-r-- 1 liteshotv3 liteshotv3  903 Mar 11 23:34 hw3posix.c
-rw-rw-r-- 1 liteshotv3 liteshotv3  903 Mar 11 23:34 hw3posix.c~
myshell>diff hw3posix.c hw3posix
Binary files hw3posix.c and hw3posix differ
myshell>exit
```