

HW-7-Project-6-Shell-Part-1

CS 503/571 Systems Basics

100 Points

Summary

Use the code in CSAPP2, 8.4.6 "Using fork and execve to Run Programs" as a starting point to develop and test a small version of a shell that forks processes in the foreground and/or the background.

What To Do

Follow the directions in BLS "How the Shell Works" lecture.

Start with program shellex.c from figures 8.22-8.24 and create your own shell by adding all the features described in the BLS lecture: read and parse command line, run command in foreground or background, implement redirection, implement pipe.

Notes:

Note 1: Although a real shell can accept a command line with an arbitrary number of pipes, in this homework it is sufficient to implement just one pipe. So, your shell should be able to handle correctly a command line like

command-1 | command-2 < input-file > output-file &

Notice that the placement of "< input-file" and "> output-file" in the command line is not critical since "< input-file" redirects the input of command-1 and "> output-file" redirects the output of command-2. If there is no pipe, "> output-file" redirects the output of command-1.

Note 2: Your shell program can execute programs in the current directory but does not inherit PATH automatically, so

> ls

does not work, but

> /usr/bin/ls

works

and, this works (when myspin is in the current directory)

> myspin 10 &

Note 3: Files csapp.c, csapp.h, shellex.c, myspin.c are in FILES/PROGRAMS

To compile and run shellex.c do the following

gcc -c csapp.c

gcc -o shellex shellex.c csapp.o -lpthread


```

/* $begin shellmain */
#include "csapp.h"

#define MAXARGS 128

/* function prototypes */
void eval(char*cmdline);
int parseline(char *buf, char **argv);
int builtin_command(char **argv);

int main()
{
    char cmdline[MAXLINE]; /* command line */

    while (1) {
        /* read */
        printf("> ");
        Fgets(cmdline, MAXLINE, stdin);
        if (feof(stdin))
            exit(0);

        /* evaluate */
        eval(cmdline);
    }
}

/* $end shellmain */

/* $begin eval */
/* eval - evaluate a command line */
void eval(char *cmdline)
{
    char *argv[MAXARGS]; /* argv for execve() */
    char buf[MAXLINE]; /* holds modified command line */
    int bg; /* should the job run in bg or fg? */
    pid_t pid; /* process id */

    strcpy(buf, cmdline);
    bg = parseline(buf, argv);
    if (argv[0] == NULL)
        return; /* ignore empty lines */

    if (!builtin_command(argv)) {
        if ((pid = Fork()) == 0) { /* child runs user job */
            if (execve(argv[0], argv, environ) < 0) {
                printf("%s: Command not found.\n", argv[0]);
                exit(0);
            }
        }
    }
}

```

```

        /* parent waits for foreground job to terminate */
        if (!bg) {
            int status;
            if (waitpid(pid, &status, 0) < 0)
                unix_error("waitfg: waitpid error");
        }
        else
            printf("%d %s", pid, cmdline);
    }
    return;
}

/* if first arg is a builtin command, run it and return true */
int builtin_command(char **argv)
{
    if (!strcmp(argv[0], "quit")) /* quit command */
        exit(0);
    if (!strcmp(argv[0], "&")) /* ignore singleton & */
        return 1;
    return 0; /* not a builtin command */
}

/* $end eval */

/* $begin parseline */
/* parseline - parse the command line and build the argv array */
int parseline(char *buf, char **argv)
{
    char *delim; /* points to first space delimiter */
    int argc; /* number of args */
    int bg; /* background job? */

    buf[strlen(buf)-1] = ' '; /* replace trailing '\n' with space */
    while (*buf && (*buf == ' ')) /* ignore leading spaces */
        buf++;

    /* build the argv list */
    argc = 0;
    while ((delim = strchr(buf, ' '))) {
        argv[argc++] = buf;
        *delim = '\0';
        buf = delim + 1;
        while (*buf && (*buf == ' ')) /* ignore spaces */
            buf++;
    }
    argv[argc] = NULL;

    if (argc == 0) /* ignore blank line */

```

```
        return 1;

    /* should the job run in the background? */
    if ((bg = (*argv[argc-1] == '&')) != 0)
        argv[--argc] = NULL;

    return bg;
}
/* $end parseline */
```