```
Yang Zheng
CS347 Spring23
Assignment3
```

```
/* CSCI347 Spring23
 * Assignment 2
 * Modified April 11, 2023 Yang zheng
 */

#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <errno.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <stdbool.h>
#include "defn.h"

/* Constants */

#define LINELEN 1024
int args = 0;
int shift = 0;
int arg_count = 0;
char** command_line = NULL;

/* Prototypes */

void processline (char *line);

void off_quote(char *line) {
  int j = 0;
  int lineLength = strlen(line);
  for (int i = 0; i < lineLength; i++) {
    if (line[i] != '"') {
        line[j++] = line[i];
    }
  }
  line[j] = '\0';
}

/* find the comment and get rid of the comment */
void off_comment(char *line) {
  char* start = line;
  while (*start != '\0') {
    if (*start == '#' && *(start - 1) != '$') {
      *start = '\0';
      break;
    }
    start++;
  }
}

bool is_empty_or_spaces(char *line) {
    int i = 0;
    while (line[i] != '\0') {
        if (line[i] != ' ' && line[i] != '\t' && line[i] != '\n') {
            return false;  // found non-space character, line is not empty or fu
ll of spaces
        }
        i++;
    }
    return true;  // end of line reached without finding non-space character, line is empty or full of spaces
}

char** arg_parse (char *line, int *argcptr) {
  int count = 1;
  int i = 0;
  bool no_quote = true;
  int length = strlen(line);

  while (line[i] != 0 && i < length) {
    if (line[i] != ' ') {
      while (line[i] != 0 && i < length) {
              if (line[i] == '"') {
          no_quote = !no_quote;
        }
        if (line[i] == ' ') {
          if (no_quote == false) { // if we have read a \", don't do anything
            ;
          } else {
            count++;
            break;
          }
        }
        i++;
      }
      i++;
    } else {
      i++;
    }
  }

  if (no_quote == false) {
    fprintf(stderr, "No matching double quotes");
  }

  i = 0;
  int j = 0;
  char** arr = (char**) malloc ((count + 1) * sizeof(char*));
  if (arr == NULL) {
     fprintf (stderr, "Failed to malloc");
  }

  while (line[i] != 0 && i < length) {
    if (line[i] != ' ') {
      arr[j] = &line[i];
      j++;
      while (line[i] != 0 && i < length) {
              if (line[i] == '"') {
          no_quote = !no_quote;
        }
        if (line[i] == ' ') {
          if (no_quote == false) { // if we have read a \", don't do anything
            ;
          } else {
            line[i] = 0;
            break;
          }
        }
        i++;
      }
      i++;
    } else {
      i++;
    }
  }

  for (int i = 0; i < j; i++) {
    off_quote(arr[i]);
  }

  arr[count] = NULL;
  *argcptr = count;

  // for (int i = 0; i < count; i++) {
  //   printf("arr[%d]: %s\n", i, arr[i]);
  // }
  return arr;
}


/* Shell main */
int
main (int argc, char **argv)
{
  // for (int i = 0; i < argc; i++) {
  //   printf("argv[%d]: %s\n", i, argv[i]);
  // }
  arg_count = argc - 1;
  args = argc - 1; //  args starts from index 2 to index n - 1 of the command li
ne
  command_line = argv;
  char buffer[LINELEN];
  int len;
  FILE* read;
  if (argc == 1) {
    read = stdin;
  } else {
    // char* filename = argv[1];
    read = fopen(argv[1], "r");
    if (read == NULL) {
      fprintf(stderr, "Failed to open file %s\n", argv[1]);
      exit(127);
    }
  }
  while (1) {

    /* prompt and get line */
    if (read == stdin) {
      fprintf (stderr, "%% ");
    }

    if (fgets (buffer, LINELEN, read) != buffer) {
      break;
    }

    if (!is_empty_or_spaces(buffer)) {
      /* Get rid of \n at end of buffer. */
      // printf("buffer: %s\n", buffer);
      len = strlen(buffer);
      if (buffer[len-1] == '\n')
          buffer[len-1] = 0;
      off_comment(buffer);
      /* Run it ... */
      processline (buffer);
    }
    if (feof(read)) {
      break;
    }
  }

  if (!feof(read)) {
    perror ("read");
  }

  fclose(read);
  return 0;             /* Also known as exit (0); */
}


void processline (char *line)
{
    pid_t  cpid;
    int    status;

    char newLine[LINELEN] = {0};
    int condition = expand(line, newLine, LINELEN);
    // printf("newLine: %s\n", newLine);
    if (condition == -1) { // if expand failed, print error message
      fprintf(stderr, "Expand failed\n");
      return;
    }

    int argc = 0;
    char** p_arr = arg_parse(newLine, &argc);
    // printf("p_arr[0]: %s\n", p_arr[0]);
    if (newLine == NULL || p_arr[0] == NULL) {
      return;
    }

    /* check if new line contains builtin command before fork */
    if (exec_builtin(p_arr) < 0) {
      /* Start a new process to do the job. */
      cpid = fork();
      if (cpid < 0) {
        /* Fork wasn't successful */
        perror ("fork");
        return;
      }

      /* Check for who we are! */
      if (cpid == 0) {
        /* We are the child! */
        // printf("p_arr[0]: %s\n", p_arr[0]);
        execvp(p_arr[0], p_arr);

        /* execlp reurned, wasn't successful */
        perror ("exec");
        fclose(stdin);  // avoid a linux stdio bug
        exit (127);
      }

      /* free pointer array */
      free(p_arr);
      p_arr = NULL;

      /* Have the parent wait for child to complete */
      if (wait (&status) < 0) {
        /* Wait wasn't successful */
        perror ("wait");
      }
    } else {
      // free(p_arr);
      // p_arr = NULL;
      ;
    }

}
```

```
#include <stdio.h>
#include <string.h>
#include <stdbool.h>
#include <stdlib.h>
#include <unistd.h>
#include <ctype.h>
#include <dirent.h>
#include "defn.h"

int result = 0; //  result of expand

void cat(char* new, char* to_cat, int* space) {
    // printf("space: %d, to_cat: %d, new: %d\n", *space, strlen(to_cat), strlen
(new));
    if (strlen(to_cat) + strlen(new) <= *space) {
        strcat(new, to_cat);
        *space -= strlen(to_cat);
    } else {
        fprintf(stderr, "No enough space to add\n");
    }
}

int expand (char *orig, char *new, int newsize) {
    // need a pointer points to the first char of NAME
    char *name = orig;

    // another pointer finds the first '}' and set it to '\0'
    char *end = orig;
    char* value = 0; // the value of the environment variable
    char pid_str[16] = {0};
    int space = newsize;
    bool has_quote = false; //  if we read a ${, we set it to true
    // printf("orig: %s\n", orig);
    if (orig[5] == '}') {
        printf("fuck\n");
    }
    while (*name != '\0' && *end != '\0') {
        while (*name != '{') {
            if (*name == '\0') { //  if we never read a {
                if (new[strlen(new) - 1] == ' ') {
                    new[strlen(new) - 1] = '\0';
                }
                return result;
            }
            if (*name == '$'){
                name++;
                if (*name == '$') { //  this will increment name
                    if (sprintf(pid_str, "%d", getpid()) >= 0) {
                        cat(new, pid_str, &space);
                    } else {
                        fprintf(stderr, "failed to get pid");
                        result = -1;
                        return result;
                    }
                } else if (*name == '{') {
                    has_quote = !has_quote;
                    break;
                } else if (isdigit(*name)) {
                    char num[10] = {0};
                    if (args > 0) {
                        while (isdigit(*name)) {
                            char n = *name;
                            strcat(num, &n);
                            name++;
                        }
                        int pattern_n = atoi(num);
                        if (pattern_n >= args) {
                            cat(new, "", &space);
                        } else {
                            cat(new, command_line[pattern_n + 1 + shift], &space
);  //  out of bounds?
                        }
                        name--;
                    } else { // interactive mode
                        if (atoi(num) == 0) {
                            cat(new, "./ush", &space);
                        } else {
                            cat(new, "", &space);
                        }
                    }
                } else if (*name == ’#’) {
                    char pound[3] = {0};
                    if (args > 0) {
                        if (sprintf(pound, "%d", args) >= 0) {
                            cat(new, pound, &space);
                        } else {
                            fprintf(stderr, "failed to get #");
                            result = -1;
                            return result;
                        }
                    } else {
                        cat(new, "1", &space);
                    }
                } else { //  if we read a $ that is not a ${ or $$, we do nothin
g
                    name--;
                    cat(new, name, &space);
                    return result;
                }
            } else if (*name == ’*’) {
                end = (name + 1);
                char* r_express = (name + 1);
                DIR *dir;
                struct dirent *ent;
                dir = opendir(".");
                bool reached_end = false;
                if (*end == ’ ’ || *end == ’\0’) { //  if there is no pattern
                    r_express = "";
                } else {
                    while (*end != ’ ’ && *end != ’\0’) {
                        end++;
                    }
                    if (*end == ’ ’) {
                        *end = ’\0’;
                    } else {
                        reached_end = true;
                    }
                }
                if (dir != NULL) {
                    bool matched = false;
                    if (strchr(r_express, ’/’) != NULL) {
                            fprintf(stderr, "can’t include /\n");
                            result = -1;
                            return result;
                    }
                    while ((ent = readdir(dir)) != NULL) {
                        if (strcmp(ent->d_name + strlen(ent->d_name) - strlen(r_
express), r_express) == 0
                        && ent->d_name[0] != ’.’) {
                            matched = true;
                            cat(new, ent->d_name, &space);
                            cat(new, " ", &space);
                        }
                    }
                    if (matched == false) { //  if we can’t find matching files
                        cat(new, r_express, &space);
                    }
                    closedir(dir);
                } else {
                    perror("Failed to open directory");
                    result = -1;
                    return result;
                }
                if (reached_end) {
                    if (new[strlen(new) - 1] == ’ ’) {
                        new[strlen(new) - 1] = ’\0’;
                    }
                    // printf("here\n");
                    break;
                } else {
                    name = end;
                    *end = ’ ’;

                }
            } else if (*name == ’\\’) {
                if (*(name + 1) == ’*’) {
                    cat(new, "*", &space);
                }
                while (*name != ’ ’ && *name != ’\0’) {
                    name++;
                }
                if (*name == ’\0’) {
                    break;
                }
            } else {
                char append[1] = {0};
                append[0] = orig[name - orig];
                append[1] = ’\0’;
                cat(new, append, &space);
                if (*name != ’ ’ && *(name + 1) == ’*’) {
                    cat(new, "*", &space);
                    name++;
                }

            }
            name++;
```

```
        }
        // printf("end: %c\n", *end);
        // printf("break\n");
        name++;
        // end = name;
        //set the last char of orig to '\0', now name points to a string
        if (has_quote == true) {
            while (*end != '}') {
                printf("end is at: %ld\n", end - orig);
                if (*end == '\0') {
                    fprintf(stderr, "Error: missing '}'\n");
                    result = -1;
                    return result;
                }
                end++;
            }
            *end = '\0';
            value = getenv(name);
            if (value == NULL) {
                cat(new, "", &space);
            } else {
                cat(new, value, &space);
            }
            *end = '}'; // set it back to '}
            end++;
            name = end;
        }
    }
    result = 1;
    return result;
}
```

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <errno.h>
#include <unistd.h>
#include <sys/stat.h>
#include <pwd.h>
#include <grp.h>
#include <time.h>
#include "defn.h"

static char* list[] = {"exit", "envset", "envunset", "cd", "shift", "unshift", "sstat"};
typedef void (*funcPtr) ();
static int is_builtin;
static char** command;

void exec_exit() {
    if (command[1] == NULL) {
        free(command);
        command = NULL;
        exit(0);
    } else {
        int exit_value = atoi(command[1]);
        free(command);
        command = NULL;
        if (exit_value == 0) {
            fprintf(stderr, "not given a valid exit value");
            // is_builtin = -1;
            // return is_builtin;
        }
        exit(exit_value);
    }
}

void exec_envset() {
    char* new_value = command[2];
    int ret = setenv(command[1], new_value, 1);
     if (ret != 0) {
        perror("setenv");
        return;
    }
}

void exec_envunset() {
    if (unsetenv(command[1]) == -1) {
        perror("envunset");
        return;
    }
}

void exec_cd() {
    int result = 0;
    if (command[1] == NULL) {
        result = chdir(getenv("HOME"));
    } else {
        result = chdir(command[1]);
    }
    if (result != 0) {
        perror("chdir");
```

```c
        // is_builtin = -1;
        // return is_builtin;
    }
}

void exec_shift() {
    if (command[1] == NULL) {
        shift = 1;
    } else {
        shift = atoi(command[1]);
    }
    if ((args - shift) < 0) {
        fprintf(stderr, "can’t shift that many arguments\n");
        // is_builtin = -1;
        // return is_builtin;
    } else {
        args = args - shift;
    }
}

void exec_unshift() {
    if (command[1] != NULL) { //  if we were given the unshift value
        if (atoi(command[1]) > shift) {
            fprintf(stderr, "can’t unshift that many arguments\n");
            // is_builtin = -1;
            // return is_builtin;
        }
        args += atoi(command[1]);
        shift -= atoi(command[1]);
    } else {
        args = arg_count;
        shift = 0;
    }

}
void exec_sstat() {
    char perms[11];
    struct stat st;
    for (int i = 1; i < sizeof(command); i++) {
        if (stat(command[i], &st) == 0) {
            printf("%s ", command[i]); //  print file name

            struct passwd *pwd = getpwuid(st.st_uid);
            if (pwd == NULL) { //  print user name
                printf("%u ", st.st_uid);
            } else {
                printf("%s ", pwd->pw_name);
            }

            struct group *grp = getgrgid(st.st_gid); // print group name
            if (grp == NULL) {
                printf("%u ", st.st_gid);
            } else {
                printf("%s ", grp->gr_name);
            }

            strmode(st.st_mode, perms); //  print permission
            printf("%s ", perms);
            printf("%lu ", st.st_nlink); //  print number of links"
            printf("%lu ", st.st_size); //  print size
            printf("%s\n", asctime(localtime(&st.st_mtime))); //  print last mod
ified time
        }
    }

}

int exec_builtin(char** line) {
    funcPtr flist[] = {exec_exit, exec_envset, exec_envunset, exec_cd, exec_shif
t, exec_unshift, exec_sstat};
    command = line;
    is_builtin = 1;
    for (int i = 0; i < sizeof(list)/sizeof(list[0]); i++) {
        if (strcmp(command[0], list[i]) == 0) {
            flist[i]();
            // is_builtin = 1;
            free(command);
            command = NULL;
            return is_builtin;
        }
    }
    /* didn't find a builtin command */
    is_builtin = -1;
    return is_builtin;
}
```

```c
#include <sys/stat.h>

int expand (char *orig, char *new, int newsize);
int exec_builtin(char** line);
void strmode(mode_t mode, char *p);
extern int args;
extern int shift;
extern int arg_count;
extern char** command_line;
```

```
Script started on 2023-05-01 12:19:02-07:00 [TERM="xterm-256color" TTY="/dev/pts
/0" COLUMNS="190" LINES="17"]
^[[?2004h^[]0;zhengy@cf162-07: ~/csci347/csci347_s23/ush^G^[[01;32mzhengy@cf162-
07^[[00m:^[[01;34m~/csci347/csci347_s23/ush^[[00m$ ./ush
^[[?2004l
% echo #this is my own test

% echo *
strmode.c expand.o 3adc test_script a2report.pdf 3.h 4a?c builtin.c a2.pdf scr4.
txt subdir a2_test 2acc d.cc printArg.c~ e.b report.ps~ b.c Makefile ush.o f.q a
1.ps printArg test 1abc own_test fully report.ps a2.ps a1.ps~ a.c c..c a2.ps~ st
rmode.o expand.c printArg.c builtin.o report.pdf aaaaaaa.c script-nq #ush.c#~ us
h ush.c header.txt a2report.ps showshift.txt test.c testa2 defn.h a1.pdf
% echo *.c *.o
strmode.c builtin.c b.c a.c c..c expand.c printArg.c aaaaaaa.c ush.c test.c expa
nd.o ush.o strmode.o builtin.o
% sstat showshift.txt
showshift.txt zhengy grp.csci.Students -rw-r--r--  1 414 Sun Apr 30 23:39:57 202
3

% echo a*
a*
% eho^H ^H^H ^Hcho c*
c*
% echo \*
*
% echo ?^H ^H/*
/*
% echo */
can’t include /
Expand failed
% ss^H ^H^H ^H^C
^[[?2004h^[]0;zhengy@cf162-07: ~/csci347/csci347_s23/ush^G^[[01;32mzhengy@cf162-
07^[[00m:^[[01;34m~/csci347/csci347_s23/ush^[[00m$ ./ush showshift.txt a b c d
^H^[[Ke f
^[[?2004l
showshift is named showshift.txt
Number of arguments is 7.
Argument 1 is a.
Argument 2 is b.
Argument 3 is c.
Argument 4 is d.
Number of arguments is 4.
Argument 1 is d.
Argument 2 is e.
Argument 3 is f.
Argument 4 is .
Number of arguments is 5.
Argument 1 is c.
Number of arguments is 7.
Now a is Argument 1.
^[[?2004h^[]0;zhengy@cf162-07: ~/csci347/csci347_s23/ush^G^[[01;32mzhengy@cf162-
07^[[00m:^[[01;34m~/csci347/csci347_s23/ush^[[00m$ exit
^[[?2004l
exit

Script done on 2023-05-01 12:20:54-07:00 [COMMAND_EXIT_CODE="0"]
Script started on 2023-04-17 12:15:08-07:00 [TERM="xterm-256color" TTY="/dev/pts
/6" COLUMNS="80" LINES="24"]
^[[?2004h^[]0;zhengy@cf162-03: ~/csci347/csci347_s23/ush^G^[[01;32mzhengy@cf162-
03^[[00m:^[[01;34m~/csci347/csci347_s23/ush^[[00m$ make
^[[?2004l
gcc -g -Wall -c ush.c
gcc -g -Wall -c expand.c
gcc -g -Wall -c builtin.c
gcc -g -Wall -o ush ush.o expand.o builtin.o
^[[?2004h^[]0;zhengy@cf162-03: ~/csci347/csci347_s23/ush^G^[[01;32mzhengy@cf162-
03^[[00m:^[[01;34m~/csci347/csci347_s23/ush^[[00m$ ./ush
^[[?2004l
% cd /home/phil/public/csci347/testa2
% ./try -H
mkdir: cannot create directory âM-^@M-^X/home/zhengy/347_test_a2âM-^@M-^Y: File
exists
~/347_test_a2 exists, use it anyway? (y/n) y
Cloning into 'csci347_s23'...
Username for 'https://gitlab.cs.wwu.edu': zhengy
Password for 'https://zhengy@gitlab.cs.wwu.edu':
remote: Enumerating objects: 121, done.^[[K
remote: Counting objects:   1% (1/97)^[[K
remote: Counting objects:   2% (2/97)^[[K
remote: Counting objects:   3% (3/97)^[[K
remote: Counting objects:   4% (4/97)^[[K
remote: Counting objects:   5% (5/97)^[[K
remote: Counting objects:   6% (6/97)^[[K
remote: Counting objects:   7% (7/97)^[[K
remote: Counting objects:   8% (8/97)^[[K
remote: Counting objects:   9% (9/97)^[[K
remote: Counting objects:  10% (10/97)^[[K
remote: Counting objects:  11% (11/97)^[[K
remote: Counting objects:  12% (12/97)^[[K
remote: Counting objects:  13% (13/97)^[[K
remote: Counting objects:  14% (14/97)^[[K
remote: Counting objects:  15% (15/97)^[[K
remote: Counting objects:  16% (16/97)^[[K
remote: Counting objects:  17% (17/97)^[[K
remote: Counting objects:  18% (18/97)^[[K
remote: Counting objects:  19% (19/97)^[[K
remote: Counting objects:  20% (20/97)^[[K
remote: Counting objects:  21% (21/97)^[[K
remote: Counting objects:  22% (22/97)^[[K
remote: Counting objects:  23% (23/97)^[[K
remote: Counting objects:  24% (24/97)^[[K
remote: Counting objects:  25% (25/97)^[[K
remote: Counting objects:  26% (26/97)^[[K
remote: Counting objects:  27% (27/97)^[[K
remote: Counting objects:  28% (28/97)^[[K
remote: Counting objects:  29% (29/97)^[[K
remote: Counting objects:  30% (30/97)^[[K
remote: Counting objects:  31% (31/97)^[[K
remote: Counting objects:  32% (32/97)^[[K
remote: Counting objects:  34% (33/97)^[[K
remote: Counting objects:  35% (34/97)^[[K
remote: Counting objects:  36% (35/97)^[[K
remote: Counting objects:  37% (36/97)^[[K
remote: Counting objects:  38% (37/97)^[[K
remote: Counting objects:  39% (38/97)^[[K
```

```
remote: Counting objects:  40% (39/97)^[[K
remote: Counting objects:  41% (40/97)^[[K
remote: Counting objects:  42% (41/97)^[[K
remote: Counting objects:  43% (42/97)^[[K
remote: Counting objects:  44% (43/97)^[[K
remote: Counting objects:  45% (44/97)^[[K
remote: Counting objects:  46% (45/97)^[[K
remote: Counting objects:  47% (46/97)^[[K
remote: Counting objects:  48% (47/97)^[[K
remote: Counting objects:  49% (48/97)^[[K
remote: Counting objects:  50% (49/97)^[[K
remote: Counting objects:  51% (50/97)^[[K
remote: Counting objects:  52% (51/97)^[[K
remote: Counting objects:  53% (52/97)^[[K
remote: Counting objects:  54% (53/97)^[[K
remote: Counting objects:  55% (54/97)^[[K
remote: Counting objects:  56% (55/97)^[[K
remote: Counting objects:  57% (56/97)^[[K
remote: Counting objects:  58% (57/97)^[[K
remote: Counting objects:  59% (58/97)^[[K
remote: Counting objects:  60% (59/97)^[[K
remote: Counting objects:  61% (60/97)^[[K
remote: Counting objects:  62% (61/97)^[[K
remote: Counting objects:  63% (62/97)^[[K
remote: Counting objects:  64% (63/97)^[[K
remote: Counting objects:  65% (64/97)^[[K
remote: Counting objects:  67% (65/97)^[[K
remote: Counting objects:  68% (66/97)^[[K
remote: Counting objects:  69% (67/97)^[[K
remote: Counting objects:  70% (68/97)^[[K
remote: Counting objects:  71% (69/97)^[[K
remote: Counting objects:  72% (70/97)^[[K
remote: Counting objects:  73% (71/97)^[[K
remote: Counting objects:  74% (72/97)^[[K
remote: Counting objects:  75% (73/97)^[[K
remote: Counting objects:  76% (74/97)^[[K
remote: Counting objects:  77% (75/97)^[[K
remote: Counting objects:  78% (76/97)^[[K
remote: Counting objects:  79% (77/97)^[[K
remote: Counting objects:  80% (78/97)^[[K
remote: Counting objects:  81% (79/97)^[[K
remote: Counting objects:  82% (80/97)^[[K
remote: Counting objects:  83% (81/97)^[[K
remote: Counting objects:  84% (82/97)^[[K
remote: Counting objects:  85% (83/97)^[[K
remote: Counting objects:  86% (84/97)^[[K
remote: Counting objects:  87% (85/97)^[[K
remote: Counting objects:  88% (86/97)^[[K
remote: Counting objects:  89% (87/97)^[[K
remote: Counting objects:  90% (88/97)^[[K
remote: Counting objects:  91% (89/97)^[[K
remote: Counting objects:  92% (90/97)^[[K
remote: Counting objects:  93% (91/97)^[[K
remote: Counting objects:  94% (92/97)^[[K
remote: Counting objects:  95% (93/97)^[[K
remote: Counting objects:  96% (94/97)^[[K
remote: Counting objects:  97% (95/97)^[[K
remote: Counting objects:  98% (96/97)^[[K
remote: Counting objects: 100% (97/97)^[[K
```

```
remote: Counting objects: 100% (97/97), done.^[[K
remote: Compressing objects:   1% (1/94)^[[K
remote: Compressing objects:   2% (2/94)^[[K
remote: Compressing objects:   3% (3/94)^[[K
remote: Compressing objects:   4% (4/94)^[[K
remote: Compressing objects:   5% (5/94)^[[K
remote: Compressing objects:   6% (6/94)^[[K
remote: Compressing objects:   7% (7/94)^[[K
remote: Compressing objects:   8% (8/94)^[[K
remote: Compressing objects:   9% (9/94)^[[K
remote: Compressing objects:  10% (10/94)^[[K
remote: Compressing objects:  11% (11/94)^[[K
remote: Compressing objects:  12% (12/94)^[[K
remote: Compressing objects:  13% (13/94)^[[K
remote: Compressing objects:  14% (14/94)^[[K
remote: Compressing objects:  15% (15/94)^[[K
remote: Compressing objects:  17% (16/94)^[[K
remote: Compressing objects:  18% (17/94)^[[K
remote: Compressing objects:  19% (18/94)^[[K
remote: Compressing objects:  20% (19/94)^[[K
remote: Compressing objects:  21% (20/94)^[[K
remote: Compressing objects:  22% (21/94)^[[K
remote: Compressing objects:  23% (22/94)^[[K
remote: Compressing objects:  24% (23/94)^[[K
remote: Compressing objects:  25% (24/94)^[[K
remote: Compressing objects:  26% (25/94)^[[K
remote: Compressing objects:  27% (26/94)^[[K
remote: Compressing objects:  28% (27/94)^[[K
remote: Compressing objects:  29% (28/94)^[[K
remote: Compressing objects:  30% (29/94)^[[K
remote: Compressing objects:  31% (30/94)^[[K
remote: Compressing objects:  32% (31/94)^[[K
remote: Compressing objects:  34% (32/94)^[[K
remote: Compressing objects:  35% (33/94)^[[K
remote: Compressing objects:  36% (34/94)^[[K
remote: Compressing objects:  37% (35/94)^[[K
remote: Compressing objects:  38% (36/94)^[[K
remote: Compressing objects:  39% (37/94)^[[K
remote: Compressing objects:  40% (38/94)^[[K
remote: Compressing objects:  41% (39/94)^[[K
remote: Compressing objects:  42% (40/94)^[[K
remote: Compressing objects:  43% (41/94)^[[K
remote: Compressing objects:  44% (42/94)^[[K
remote: Compressing objects:  45% (43/94)^[[K
remote: Compressing objects:  46% (44/94)^[[K
remote: Compressing objects:  47% (45/94)^[[K
remote: Compressing objects:  48% (46/94)^[[K
remote: Compressing objects:  50% (47/94)^[[K
remote: Compressing objects:  51% (48/94)^[[K
remote: Compressing objects:  52% (49/94)^[[K
remote: Compressing objects:  53% (50/94)^[[K
remote: Compressing objects:  54% (51/94)^[[K
remote: Compressing objects:  55% (52/94)^[[K
remote: Compressing objects:  56% (53/94)^[[K
remote: Compressing objects:  57% (54/94)^[[K
remote: Compressing objects:  58% (55/94)^[[K
remote: Compressing objects:  59% (56/94)^[[K
remote: Compressing objects:  60% (57/94)^[[K
remote: Compressing objects:  61% (58/94)^[[K
```

```
remote: Compressing objects:  62% (59/94)^[[K
remote: Compressing objects:  63% (60/94)^[[K
remote: Compressing objects:  64% (61/94)^[[K
remote: Compressing objects:  65% (62/94)^[[K
remote: Compressing objects:  67% (63/94)^[[K
remote: Compressing objects:  68% (64/94)^[[K
remote: Compressing objects:  69% (65/94)^[[K
remote: Compressing objects:  70% (66/94)^[[K
remote: Compressing objects:  71% (67/94)^[[K
remote: Compressing objects:  72% (68/94)^[[K
remote: Compressing objects:  73% (69/94)^[[K
remote: Compressing objects:  74% (70/94)^[[K
remote: Compressing objects:  75% (71/94)^[[K
remote: Compressing objects:  76% (72/94)^[[K
remote: Compressing objects:  77% (73/94)^[[K
remote: Compressing objects:  78% (74/94)^[[K
remote: Compressing objects:  79% (75/94)^[[K
remote: Compressing objects:  80% (76/94)^[[K
remote: Compressing objects:  81% (77/94)^[[K
remote: Compressing objects:  82% (78/94)^[[K
remote: Compressing objects:  84% (79/94)^[[K
remote: Compressing objects:  85% (80/94)^[[K
remote: Compressing objects:  86% (81/94)^[[K
remote: Compressing objects:  87% (82/94)^[[K
remote: Compressing objects:  88% (83/94)^[[K
remote: Compressing objects:  89% (84/94)^[[K
remote: Compressing objects:  90% (85/94)^[[K
remote: Compressing objects:  91% (86/94)^[[K
remote: Compressing objects:  92% (87/94)^[[K
remote: Compressing objects:  93% (88/94)^[[K
remote: Compressing objects:  94% (89/94)^[[K
remote: Compressing objects:  95% (90/94)^[[K
remote: Compressing objects:  96% (91/94)^[[K
remote: Compressing objects:  97% (92/94)^[[K
remote: Compressing objects:  98% (93/94)^[[K
remote: Compressing objects: 100% (94/94)^[[K
remote: Compressing objects: 100% (94/94), done.^[[K
Receiving objects:   0% (1/121)
Receiving objects:   1% (2/121)
Receiving objects:   2% (3/121)
Receiving objects:   3% (4/121)
Receiving objects:   4% (5/121)
Receiving objects:   5% (7/121)
Receiving objects:   6% (8/121)
Receiving objects:   7% (9/121)
Receiving objects:   8% (10/121)
Receiving objects:   9% (11/121)
Receiving objects:  10% (13/121)
Receiving objects:  11% (14/121)
Receiving objects:  12% (15/121)
Receiving objects:  13% (16/121)
Receiving objects:  14% (17/121)
Receiving objects:  15% (19/121)
Receiving objects:  16% (20/121)
Receiving objects:  17% (21/121)
Receiving objects:  18% (22/121)
Receiving objects:  19% (23/121)
Receiving objects:  20% (25/121)
Receiving objects:  21% (26/121)
Receiving objects:  22% (27/121)
Receiving objects:  23% (28/121)
Receiving objects:  24% (30/121)
Receiving objects:  25% (31/121)
Receiving objects:  26% (32/121)
Receiving objects:  27% (33/121)
Receiving objects:  28% (34/121)
Receiving objects:  29% (36/121)
Receiving objects:  30% (37/121)
Receiving objects:  31% (38/121)
Receiving objects:  32% (39/121)
Receiving objects:  33% (40/121)
Receiving objects:  34% (42/121)
Receiving objects:  35% (43/121)
Receiving objects:  36% (44/121)
Receiving objects:  37% (45/121)
Receiving objects:  38% (46/121)
Receiving objects:  39% (48/121)
Receiving objects:  40% (49/121)
Receiving objects:  41% (50/121)
Receiving objects:  42% (51/121)
Receiving objects:  43% (53/121)
Receiving objects:  44% (54/121)
Receiving objects:  45% (55/121)
Receiving objects:  46% (56/121)
Receiving objects:  47% (57/121)
Receiving objects:  48% (59/121)
Receiving objects:  49% (60/121)
Receiving objects:  50% (61/121)
Receiving objects:  51% (62/121)
Receiving objects:  52% (63/121)
Receiving objects:  53% (65/121)
Receiving objects:  54% (66/121)
Receiving objects:  55% (67/121)
Receiving objects:  56% (68/121)
Receiving objects:  57% (69/121)
Receiving objects:  58% (71/121)
Receiving objects:  59% (72/121)
Receiving objects:  60% (73/121)
Receiving objects:  61% (74/121)
Receiving objects:  62% (76/121)
Receiving objects:  63% (77/121)
Receiving objects:  64% (78/121)
Receiving objects:  65% (79/121)
Receiving objects:  66% (80/121)
Receiving objects:  67% (82/121)
Receiving objects:  68% (83/121)
Receiving objects:  69% (84/121)
Receiving objects:  70% (85/121)
Receiving objects:  71% (86/121)
Receiving objects:  72% (88/121)
Receiving objects:  73% (89/121)
Receiving objects:  74% (90/121)
Receiving objects:  75% (91/121)
Receiving objects:  76% (92/121)
Receiving objects:  77% (94/121)
Receiving objects:  78% (95/121)
Receiving objects:  79% (96/121)
Receiving objects:  80% (97/121)
Receiving objects:  81% (99/121)
Receiving objects:  82% (100/121)
Receiving objects:  83% (101/121)
Receiving objects:  84% (102/121)
Receiving objects:  85% (103/121)
Receiving objects:  86% (105/121)
Receiving objects:  87% (106/121)
Receiving objects:  88% (107/121)
remote: Total 121 (delta 47), reused 0 (delta 0), pack-reused 24^[[K
Receiving objects:  89% (108/121)
Receiving objects:  90% (109/121)
Receiving objects:  91% (111/121)
Receiving objects:  92% (112/121)
Receiving objects:  93% (113/121)
Receiving objects:  94% (114/121)
Receiving objects:  95% (115/121)
Receiving objects:  96% (117/121)
Receiving objects:  97% (118/121)
Receiving objects:  98% (119/121)
Receiving objects:  99% (120/121)
Receiving objects: 100% (121/121)
Receiving objects: 100% (121/121), 110.77 KiB | 2.70 MiB/s, done.
Resolving deltas:   0% (0/53)
Resolving deltas:   1% (1/53)
Resolving deltas:   3% (2/53)
Resolving deltas:   5% (3/53)
Resolving deltas:   7% (4/53)
Resolving deltas:   9% (5/53)
Resolving deltas:  11% (6/53)
Resolving deltas:  13% (7/53)
Resolving deltas:  15% (8/53)
Resolving deltas:  16% (9/53)
Resolving deltas:  18% (10/53)
Resolving deltas:  20% (11/53)
Resolving deltas:  22% (12/53)
Resolving deltas:  24% (13/53)
Resolving deltas:  26% (14/53)
Resolving deltas:  28% (15/53)
Resolving deltas:  30% (16/53)
Resolving deltas:  32% (17/53)
Resolving deltas:  33% (18/53)
Resolving deltas:  35% (19/53)
Resolving deltas:  37% (20/53)
Resolving deltas:  39% (21/53)
Resolving deltas:  41% (22/53)
Resolving deltas:  43% (23/53)
Resolving deltas:  45% (24/53)
Resolving deltas:  47% (25/53)
Resolving deltas:  49% (26/53)
Resolving deltas:  50% (27/53)
Resolving deltas:  52% (28/53)
Resolving deltas:  54% (29/53)
Resolving deltas:  56% (30/53)
Resolving deltas:  58% (31/53)
Resolving deltas:  60% (32/53)
Resolving deltas:  62% (33/53)
Resolving deltas:  64% (34/53)
Resolving deltas:  66% (35/53)
Resolving deltas:  67% (36/53)
Resolving deltas:  69% (37/53)
Resolving deltas:  71% (38/53)
Resolving deltas:  73% (39/53)
Resolving deltas:  75% (40/53)
Resolving deltas:  77% (41/53)
Resolving deltas:  79% (42/53)
Resolving deltas:  81% (43/53)
Resolving deltas:  83% (44/53)
Resolving deltas:  84% (45/53)
Resolving deltas:  86% (46/53)
Resolving deltas:  88% (47/53)
Resolving deltas:  90% (48/53)
Resolving deltas:  92% (49/53)
Resolving deltas:  94% (50/53)
Resolving deltas:  96% (51/53)
Resolving deltas:  98% (52/53)
Resolving deltas: 100% (53/53)
Resolving deltas: 100% (53/53), done.
Branch 'a2' set up to track remote branch 'a2' from 'origin'.
Switched to a new branch 'a2'
gcc -g -Wall -c builtin.c
gcc -g -Wall -o ush ush.o expand.o builtin.o

Running ush

Script output same
Exit values correct

----- ERRS -----
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
----------------
Test Errors?
Look at checked out files? Clean? % ^[[?2004h^[]0;zhengy@cf162-03: ~/csci347/csc
i347_s23/ush^G^[[01;32mzhengy@cf162-03^[[00m:^[[01;34m~/csci347/csci347_s23/ush
^[[00m$ exit
^[[?2004l
exit

Script done on 2023-04-17 12:16:05-07:00 [COMMAND_EXIT_CODE="0"]
```