```
/* CSCI347 Spring23
 * Assignment 3
 * Modified April 18, 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 <ctype.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') {
            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, li
ne 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**) calloc ((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]);
  }
  // printf("args: %d\n", count);
  arr[count] = NULL;
  *argcptr = count;

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

```
    // printf("pid: %d\n", getpid());
    if (*buffer != '\n' && !is_empty_or_spaces(buffer)) {
      /* Get rid of \n at end of 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;
    if (line == NULL) {
      printf("line is NULL\n");
      return;
    }
    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);
    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;
      ;
    }

}
```

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

//  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;
    int result = 0;
    // 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 ${, we set it to true
    // printf("orig: %s\n", orig);
    while (*name != '\0' && *end != '\0') {
        while (*name != '{') {
            if (*name == '\0') { //  if we never read a {
                if (new[strlen(new) - 1] == ' ') { //  get rid of the trailing s
pace
                    // printf("set %s null\n", new);
                    new[strlen(new) - 1] = '\0';
                }
                return result;
            }
            if (*name == '$'){
                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 {
                            // printf("shift: %d\n", shift);
                            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, "*", &space);
                        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';
                    }
                    return result;
                } else {
                    name = end;
                    *end = ' ';

                }
            } else if (*name == '\\') {
                if (*(name + 1) == '*') {
                    cat(new, "*", &space);
                }
                name++;
                // 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++;
        }
        name++;

        //set the last char of orig to '\0', now name points to a string
        if (has_quote == true) {
            while (*end != '}') {
                if (*end == '\0') {
                    fprintf(stderr, "Error: missing '}'\n");
                    result = -1;
                    return result;
                }
                end++;
            }
            has_quote = !has_quote;
            *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;
}
```

```c
#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");
```

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

void exec_shift() {
    int cur_shift = 0;
    if (command[1] == NULL) {
        cur_shift = 1;
        shift += cur_shift;
    } else {
        cur_shift = atoi(command[1]);
        shift += cur_shift;
    }
    if ((args - shift) < 0) {
        fprintf(stderr, "can't shift that many arguments\n");
        // is_builtin = -1;
        // return is_builtin;
    } else {
        args = args - cur_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", asctime(localtime(&st.st_mtime))); //  print last modi
fied 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;
}
```

```
#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 2023-05-01 20:14:27-07:00 [TERM="xterm-256color" TTY="/dev/pts
/4" COLUMNS="130" LINES="28"]
^[[?2004h^[]0;zhengy@cf162-04: ~/csci347/csci347_s23/ush^G^[[01;32mzhengy@cf162-
04^[[00m:^[[01;34m~/csci347/csci347_s23/ush^[[00m$ cd ^[[7m/home/phil/public/csc
i347/testa2^[[27m^H^H^H^H^[[1P/home/phil/public/csci347/testa3
^[[?2004l
^[[?2004h^[]0;zhengy@cf162-04: /home/phil/public/csci347/testa3^G^[[01;32mzhengy
@cf162-04^[[00m:^[[01;34m/home/phil/public/csci347/testa3^[[00m$ ./try -H
^[[?2004l
mkdir: cannot create directory âM-^@M-^X/home/zhengy/347_test_a3âM-^@M-^Y: File
exists
~/347_test_a3 exists, use it anyway? (y/n) Y^H ^Hy
Cloning into 'csci347_s23'...
Username for 'https://gitlab.cs.wwu.edu': zhengy
Password for 'https://zhengy@gitlab.cs.wwu.edu':
remote: Enumerating objects: 336, done.^[[K
remote: Counting objects:   0% (1/312)^[[K
remote: Counting objects:   1% (4/312)^[[K
remote: Counting objects:   2% (7/312)^[[K
remote: Counting objects:   3% (10/312)^[[K
remote: Counting objects:   4% (13/312)^[[K
remote: Counting objects:   5% (16/312)^[[K
remote: Counting objects:   6% (19/312)^[[K
remote: Counting objects:   7% (22/312)^[[K
remote: Counting objects:   8% (25/312)^[[K
remote: Counting objects:   9% (29/312)^[[K
remote: Counting objects:  10% (32/312)^[[K
remote: Counting objects:  11% (35/312)^[[K
remote: Counting objects:  12% (38/312)^[[K
remote: Counting objects:  13% (41/312)^[[K
remote: Counting objects:  14% (44/312)^[[K
remote: Counting objects:  15% (47/312)^[[K
remote: Counting objects:  16% (50/312)^[[K
remote: Counting objects:  17% (54/312)^[[K
remote: Counting objects:  18% (57/312)^[[K
remote: Counting objects:  19% (60/312)^[[K
remote: Counting objects:  20% (63/312)^[[K
remote: Counting objects:  21% (66/312)^[[K
remote: Counting objects:  22% (69/312)^[[K
remote: Counting objects:  23% (72/312)^[[K
remote: Counting objects:  24% (75/312)^[[K
remote: Counting objects:  25% (78/312)^[[K
remote: Counting objects:  26% (82/312)^[[K
remote: Counting objects:  27% (85/312)^[[K
remote: Counting objects:  28% (88/312)^[[K
remote: Counting objects:  29% (91/312)^[[K
remote: Counting objects:  30% (94/312)^[[K
remote: Counting objects:  31% (97/312)^[[K
remote: Counting objects:  32% (100/312)^[[K
remote: Counting objects:  33% (103/312)^[[K
remote: Counting objects:  34% (107/312)^[[K
remote: Counting objects:  35% (110/312)^[[K
remote: Counting objects:  36% (113/312)^[[K
remote: Counting objects:  37% (116/312)^[[K
remote: Counting objects:  38% (119/312)^[[K
remote: Counting objects:  39% (122/312)^[[K
remote: Counting objects:  40% (125/312)^[[K
remote: Counting objects:  41% (128/312)^[[K
remote: Counting objects:  42% (132/312)^[[K
```

```
remote: Counting objects:  43% (135/312)^[[K
remote: Counting objects:  44% (138/312)^[[K
remote: Counting objects:  45% (141/312)^[[K
remote: Counting objects:  46% (144/312)^[[K
remote: Counting objects:  47% (147/312)^[[K
remote: Counting objects:  48% (150/312)^[[K
remote: Counting objects:  49% (153/312)^[[K
remote: Counting objects:  50% (156/312)^[[K
remote: Counting objects:  51% (160/312)^[[K
remote: Counting objects:  52% (163/312)^[[K
remote: Counting objects:  53% (166/312)^[[K
remote: Counting objects:  54% (169/312)^[[K
remote: Counting objects:  55% (172/312)^[[K
remote: Counting objects:  56% (175/312)^[[K
remote: Counting objects:  57% (178/312)^[[K
remote: Counting objects:  58% (181/312)^[[K
remote: Counting objects:  59% (185/312)^[[K
remote: Counting objects:  60% (188/312)^[[K
remote: Counting objects:  61% (191/312)^[[K
remote: Counting objects:  62% (194/312)^[[K
remote: Counting objects:  63% (197/312)^[[K
remote: Counting objects:  64% (200/312)^[[K
remote: Counting objects:  65% (203/312)^[[K
remote: Counting objects:  66% (206/312)^[[K
remote: Counting objects:  67% (210/312)^[[K
remote: Counting objects:  68% (213/312)^[[K
remote: Counting objects:  69% (216/312)^[[K
remote: Counting objects:  70% (219/312)^[[K
remote: Counting objects:  71% (222/312)^[[K
remote: Counting objects:  72% (225/312)^[[K
remote: Counting objects:  73% (228/312)^[[K
remote: Counting objects:  74% (231/312)^[[K
remote: Counting objects:  75% (234/312)^[[K
remote: Counting objects:  76% (238/312)^[[K
remote: Counting objects:  77% (241/312)^[[K
remote: Counting objects:  78% (244/312)^[[K
remote: Counting objects:  79% (247/312)^[[K
remote: Counting objects:  80% (250/312)^[[K
remote: Counting objects:  81% (253/312)^[[K
remote: Counting objects:  82% (256/312)^[[K
remote: Counting objects:  83% (259/312)^[[K
remote: Counting objects:  84% (263/312)^[[K
remote: Counting objects:  85% (266/312)^[[K
remote: Counting objects:  86% (269/312)^[[K
remote: Counting objects:  87% (272/312)^[[K
remote: Counting objects:  88% (275/312)^[[K
remote: Counting objects:  89% (278/312)^[[K
remote: Counting objects:  90% (281/312)^[[K
remote: Counting objects:  91% (284/312)^[[K
remote: Counting objects:  92% (288/312)^[[K
remote: Counting objects:  93% (291/312)^[[K
remote: Counting objects:  94% (294/312)^[[K
remote: Counting objects:  95% (297/312)^[[K
remote: Counting objects:  96% (300/312)^[[K
remote: Counting objects:  97% (303/312)^[[K
remote: Counting objects:  98% (306/312)^[[K
remote: Counting objects:  99% (309/312)^[[K
remote: Counting objects: 100% (312/312)^[[K
remote: Counting objects: 100% (312/312), done.^[[K
```

```
remote: Compressing objects:   0% (1/307)^[[K
remote: Compressing objects:   1% (4/307)^[[K
remote: Compressing objects:   2% (7/307)^[[K
remote: Compressing objects:   3% (10/307)^[[K
remote: Compressing objects:   4% (13/307)^[[K
remote: Compressing objects:   5% (16/307)^[[K
remote: Compressing objects:   6% (19/307)^[[K
remote: Compressing objects:   7% (22/307)^[[K
remote: Compressing objects:   8% (25/307)^[[K
remote: Compressing objects:   9% (28/307)^[[K
remote: Compressing objects:  10% (31/307)^[[K
remote: Compressing objects:  11% (34/307)^[[K
remote: Compressing objects:  12% (37/307)^[[K
remote: Compressing objects:  13% (40/307)^[[K
remote: Compressing objects:  14% (43/307)^[[K
remote: Compressing objects:  15% (47/307)^[[K
remote: Compressing objects:  16% (50/307)^[[K
remote: Compressing objects:  17% (53/307)^[[K
remote: Compressing objects:  18% (56/307)^[[K
remote: Compressing objects:  19% (59/307)^[[K
remote: Compressing objects:  20% (62/307)^[[K
remote: Compressing objects:  21% (65/307)^[[K
remote: Compressing objects:  22% (68/307)^[[K
remote: Compressing objects:  23% (71/307)^[[K
remote: Compressing objects:  24% (74/307)^[[K
remote: Compressing objects:  25% (77/307)^[[K
remote: Compressing objects:  26% (80/307)^[[K
remote: Compressing objects:  27% (83/307)^[[K
remote: Compressing objects:  28% (86/307)^[[K
remote: Compressing objects:  29% (90/307)^[[K
remote: Compressing objects:  30% (93/307)^[[K
remote: Compressing objects:  31% (96/307)^[[K
remote: Compressing objects:  32% (99/307)^[[K
remote: Compressing objects:  33% (102/307)^[[K
remote: Compressing objects:  34% (105/307)^[[K
remote: Compressing objects:  35% (108/307)^[[K
remote: Compressing objects:  36% (111/307)^[[K
remote: Compressing objects:  37% (114/307)^[[K
remote: Compressing objects:  38% (117/307)^[[K
remote: Compressing objects:  39% (120/307)^[[K
remote: Compressing objects:  40% (123/307)^[[K
remote: Compressing objects:  41% (126/307)^[[K
remote: Compressing objects:  42% (129/307)^[[K
remote: Compressing objects:  43% (133/307)^[[K
remote: Compressing objects:  44% (136/307)^[[K
remote: Compressing objects:  45% (139/307)^[[K
remote: Compressing objects:  46% (142/307)^[[K
remote: Compressing objects:  47% (145/307)^[[K
remote: Compressing objects:  48% (148/307)^[[K
remote: Compressing objects:  49% (151/307)^[[K
remote: Compressing objects:  50% (154/307)^[[K
remote: Compressing objects:  51% (157/307)^[[K
remote: Compressing objects:  52% (160/307)^[[K
remote: Compressing objects:  53% (163/307)^[[K
remote: Compressing objects:  54% (166/307)^[[K
remote: Compressing objects:  55% (169/307)^[[K
remote: Compressing objects:  56% (172/307)^[[K
remote: Compressing objects:  57% (175/307)^[[K
remote: Compressing objects:  58% (179/307)^[[K
```

```
remote: Compressing objects:  59% (182/307)^[[K
remote: Compressing objects:  60% (185/307)^[[K
remote: Compressing objects:  61% (188/307)^[[K
remote: Compressing objects:  62% (191/307)^[[K
remote: Compressing objects:  63% (194/307)^[[K
remote: Compressing objects:  64% (197/307)^[[K
remote: Compressing objects:  65% (200/307)^[[K
remote: Compressing objects:  66% (203/307)^[[K
remote: Compressing objects:  67% (206/307)^[[K
remote: Compressing objects:  68% (209/307)^[[K
remote: Compressing objects:  69% (212/307)^[[K
remote: Compressing objects:  70% (215/307)^[[K
remote: Compressing objects:  71% (218/307)^[[K
remote: Compressing objects:  72% (222/307)^[[K
remote: Compressing objects:  73% (225/307)^[[K
remote: Compressing objects:  74% (228/307)^[[K
remote: Compressing objects:  75% (231/307)^[[K
remote: Compressing objects:  76% (234/307)^[[K
remote: Compressing objects:  77% (237/307)^[[K
remote: Compressing objects:  78% (240/307)^[[K
remote: Compressing objects:  79% (243/307)^[[K
remote: Compressing objects:  80% (246/307)^[[K
remote: Compressing objects:  81% (249/307)^[[K
remote: Compressing objects:  82% (252/307)^[[K
remote: Compressing objects:  83% (255/307)^[[K
remote: Compressing objects:  84% (258/307)^[[K
remote: Compressing objects:  85% (261/307)^[[K
remote: Compressing objects:  86% (265/307)^[[K
remote: Compressing objects:  87% (268/307)^[[K
remote: Compressing objects:  88% (271/307)^[[K
remote: Compressing objects:  89% (274/307)^[[K
remote: Compressing objects:  90% (277/307)^[[K
remote: Compressing objects:  91% (280/307)^[[K
remote: Compressing objects:  92% (283/307)^[[K
remote: Compressing objects:  93% (286/307)^[[K
remote: Compressing objects:  94% (289/307)^[[K
remote: Compressing objects:  95% (292/307)^[[K
remote: Compressing objects:  96% (295/307)^[[K
remote: Compressing objects:  97% (298/307)^[[K
remote: Compressing objects:  98% (301/307)^[[K
remote: Compressing objects:  99% (304/307)^[[K
remote: Compressing objects: 100% (307/307)^[[K
remote: Compressing objects: 100% (307/307), done.^[[K
Receiving objects:   0% (1/336)
Receiving objects:   1% (4/336)
Receiving objects:   2% (7/336)
Receiving objects:   3% (11/336)
Receiving objects:   4% (14/336)
Receiving objects:   5% (17/336)
Receiving objects:   6% (21/336)
Receiving objects:   7% (24/336)
Receiving objects:   8% (27/336)
Receiving objects:   9% (31/336)
Receiving objects:  10% (34/336)
Receiving objects:  11% (37/336)
Receiving objects:  12% (41/336)
Receiving objects:  13% (44/336)
Receiving objects:  14% (48/336)
Receiving objects:  15% (51/336)
```

```
Receiving objects:  16% (54/336)
Receiving objects:  17% (58/336)
Receiving objects:  18% (61/336)
Receiving objects:  19% (64/336)
Receiving objects:  20% (68/336)
Receiving objects:  21% (71/336)
Receiving objects:  22% (74/336)
Receiving objects:  23% (78/336)
Receiving objects:  24% (81/336)
Receiving objects:  25% (84/336)
Receiving objects:  26% (88/336)
Receiving objects:  27% (91/336)
Receiving objects:  28% (95/336)
Receiving objects:  29% (98/336)
Receiving objects:  30% (101/336)
Receiving objects:  31% (105/336)
Receiving objects:  32% (108/336)
Receiving objects:  33% (111/336)
Receiving objects:  34% (115/336)
Receiving objects:  35% (118/336)
Receiving objects:  36% (121/336)
Receiving objects:  37% (125/336)
Receiving objects:  38% (128/336)
Receiving objects:  39% (132/336)
Receiving objects:  40% (135/336)
Receiving objects:  41% (138/336)
Receiving objects:  42% (142/336)
Receiving objects:  43% (145/336)
Receiving objects:  44% (148/336)
Receiving objects:  45% (152/336)
Receiving objects:  46% (155/336)
Receiving objects:  47% (158/336)
Receiving objects:  48% (162/336)
Receiving objects:  49% (165/336)
Receiving objects:  50% (168/336)
Receiving objects:  51% (172/336)
Receiving objects:  52% (175/336)
Receiving objects:  53% (179/336)
Receiving objects:  54% (182/336)
Receiving objects:  55% (185/336)
Receiving objects:  56% (189/336)
Receiving objects:  57% (192/336)
Receiving objects:  58% (195/336)
Receiving objects:  59% (199/336)
Receiving objects:  60% (202/336)
Receiving objects:  61% (205/336)
Receiving objects:  62% (209/336)
Receiving objects:  63% (212/336)
Receiving objects:  64% (216/336)
Receiving objects:  65% (219/336)
Receiving objects:  66% (222/336)
Receiving objects:  67% (226/336)
Receiving objects:  68% (229/336)
Receiving objects:  69% (232/336)
Receiving objects:  70% (236/336)
Receiving objects:  71% (239/336)
Receiving objects:  72% (242/336)
Receiving objects:  73% (246/336)
Receiving objects:  74% (249/336)
```

```
Receiving objects:  75% (252/336)
Receiving objects:  76% (256/336)
Receiving objects:  77% (259/336)
Receiving objects:  78% (263/336)
Receiving objects:  79% (266/336)
Receiving objects:  80% (269/336)
Receiving objects:  81% (273/336)
Receiving objects:  82% (276/336)
Receiving objects:  83% (279/336)
Receiving objects:  84% (283/336)
Receiving objects:  85% (286/336)
Receiving objects:  86% (289/336)
Receiving objects:  87% (293/336)
Receiving objects:  88% (296/336)
Receiving objects:  89% (300/336)
Receiving objects:  90% (303/336)
Receiving objects:  91% (306/336)
Receiving objects:  92% (310/336)
Receiving objects:  93% (313/336)
Receiving objects:  94% (316/336)
Receiving objects:  95% (320/336)
Receiving objects:  96% (323/336)
remote: Total 336 (delta 187), reused 0 (delta 0), pack-reused 24^[[K
Receiving objects:  97% (326/336)
Receiving objects:  98% (330/336)
Receiving objects:  99% (333/336)
Receiving objects: 100% (336/336)
Receiving objects: 100% (336/336), 361.85 KiB | 3.09 MiB/s, done.
Resolving deltas:   0% (0/193)
Resolving deltas:   1% (2/193)
Resolving deltas:   2% (4/193)
Resolving deltas:   3% (6/193)
Resolving deltas:   4% (8/193)
Resolving deltas:   5% (10/193)
Resolving deltas:   6% (12/193)
Resolving deltas:   7% (14/193)
Resolving deltas:   8% (16/193)
Resolving deltas:   9% (18/193)
Resolving deltas:  10% (20/193)
Resolving deltas:  11% (22/193)
Resolving deltas:  12% (24/193)
Resolving deltas:  13% (26/193)
Resolving deltas:  14% (28/193)
Resolving deltas:  15% (29/193)
Resolving deltas:  16% (31/193)
Resolving deltas:  17% (33/193)
Resolving deltas:  18% (35/193)
Resolving deltas:  19% (37/193)
Resolving deltas:  20% (39/193)
Resolving deltas:  21% (41/193)
Resolving deltas:  22% (43/193)
Resolving deltas:  23% (45/193)
Resolving deltas:  24% (47/193)
Resolving deltas:  25% (49/193)
Resolving deltas:  26% (51/193)
Resolving deltas:  27% (53/193)
Resolving deltas:  28% (55/193)
Resolving deltas:  29% (56/193)
Resolving deltas:  30% (58/193)
```

```
Resolving deltas:  31% (60/193)
Resolving deltas:  32% (62/193)
Resolving deltas:  33% (64/193)
Resolving deltas:  34% (66/193)
Resolving deltas:  35% (68/193)
Resolving deltas:  36% (70/193)
Resolving deltas:  37% (72/193)
Resolving deltas:  38% (74/193)
Resolving deltas:  39% (76/193)
Resolving deltas:  40% (78/193)
Resolving deltas:  41% (80/193)
Resolving deltas:  42% (82/193)
Resolving deltas:  43% (83/193)
Resolving deltas:  44% (85/193)
Resolving deltas:  45% (87/193)
Resolving deltas:  46% (89/193)
Resolving deltas:  47% (91/193)
Resolving deltas:  48% (93/193)
Resolving deltas:  49% (95/193)
Resolving deltas:  50% (97/193)
Resolving deltas:  51% (99/193)
Resolving deltas:  52% (101/193)
Resolving deltas:  53% (103/193)
Resolving deltas:  54% (105/193)
Resolving deltas:  55% (107/193)
Resolving deltas:  56% (109/193)
Resolving deltas:  57% (111/193)
Resolving deltas:  58% (112/193)
Resolving deltas:  59% (114/193)
Resolving deltas:  60% (116/193)
Resolving deltas:  61% (118/193)
Resolving deltas:  62% (120/193)
Resolving deltas:  63% (122/193)
Resolving deltas:  64% (124/193)
Resolving deltas:  65% (126/193)
Resolving deltas:  66% (128/193)
Resolving deltas:  67% (130/193)
Resolving deltas:  68% (132/193)
Resolving deltas:  69% (134/193)
Resolving deltas:  70% (136/193)
Resolving deltas:  71% (138/193)
Resolving deltas:  72% (139/193)
Resolving deltas:  73% (141/193)
Resolving deltas:  74% (143/193)
Resolving deltas:  75% (145/193)
Resolving deltas:  76% (147/193)
Resolving deltas:  77% (149/193)
Resolving deltas:  78% (151/193)
Resolving deltas:  79% (153/193)
Resolving deltas:  80% (155/193)
Resolving deltas:  81% (157/193)
Resolving deltas:  82% (159/193)
Resolving deltas:  83% (161/193)
Resolving deltas:  84% (163/193)
Resolving deltas:  85% (165/193)
Resolving deltas:  86% (166/193)
Resolving deltas:  87% (168/193)
Resolving deltas:  88% (170/193)
Resolving deltas:  89% (172/193)
```

```
Resolving deltas:  90% (174/193)
Resolving deltas:  91% (176/193)
Resolving deltas:  92% (178/193)
Resolving deltas:  93% (180/193)
Resolving deltas:  94% (182/193)
Resolving deltas:  95% (184/193)
Resolving deltas:  96% (186/193)
Resolving deltas:  97% (188/193)
Resolving deltas:  98% (190/193)
Resolving deltas:  99% (192/193)
Resolving deltas: 100% (193/193)
Resolving deltas: 100% (193/193), done.
Branch 'a3' set up to track remote branch 'a3' from 'origin'.
Switched to a new branch 'a3'
Making REF
gcc -g -Wall -c builtin.c
gcc -g -Wall -c strmode.c
gcc -g -Wall -o ush ush.o expand.o builtin.o strmode.o

run tests? y
Running ush

Script output same
Exit values correct

----- ERRS -----
% % % % % % % % % % % % % % % % %
----------------

Run error tests? y
Testing errors with /home/phil/public/csci347/testa3/ush.err and 5 arguments (sh
ould be 5)
Initial arguments are arg1, arg2, arg3, and arg4.
Doing a shift of 4 (no error)
We now have 1 argument, arg1 is ''.
Doing a shift of 4 and 1 (should be errors)
can't shift that many arguments
can't shift that many arguments
Now doing 'unshift 5' (should be an error)
Did an unshift ... number of args is 5, should be 5.
Testing sstat errors
End of error tests


Look at checked out files? y
^[[?2004h^[]0;zhengy@cf162-04: ~/347_test_a3/csci347_s23/ush^G^[[01;32mzhengy@cf
162-04^[[00m:^[[01;34m~/347_test_a3/csci347_s23/ush^[[00m$
^[[?2004l
^[[?2004h^[]0;zhengy@cf162-04: ~/347_test_a3/csci347_s23/ush^G^[[01;32mzhengy@cf
162-04^[[00m:^[[01;34m~/347_test_a3/csci347_s23/ush^[[00m$ exit
^[[?2004l
exit
Clean? y
cleaning
removing ~/347_test_a3
^[[?2004h^[]0;zhengy@cf162-04: /home/phil/public/csci347/testa3^G^[[01;32mzhengy
@cf162-04^[[00m:^[[01;34m/home/phil/public/csci347/testa3^[[00m$ exit
^[[?2004l
exit
```

```
Script done on 2023-05-01 20:15:18-07:00 [COMMAND_EXIT_CODE="0"]
```

```
/* CSCI347 Spring23
 * Assignment 3
 * Modified April 18, 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 <ctype.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') {
            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, li
ne 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**) calloc ((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]);
  }
  // printf("args: %d\n", count);
  arr[count] = NULL;
  *argcptr = count;

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

```
    // printf("pid: %d\n", getpid());
    if (*buffer != '\n' && !is_empty_or_spaces(buffer)) {
      /* Get rid of \n at end of 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;
    if (line == NULL) {
      printf("line is NULL\n");
      return;
    }
    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);
    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;
      ;
    }

}
```

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

//  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;
    int result = 0;
    // 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 ${, we set it to true
    // printf("orig: %s\n", orig);
    while (*name != '\0' && *end != '\0') {
        while (*name != '{') {
            if (*name == '\0') { //  if we never read a {
                if (new[strlen(new) - 1] == ' ') { //  get rid of the trailing s
pace
                    // printf("set %s null\n", new);
                    new[strlen(new) - 1] = '\0';
                }
                return result;
            }
            if (*name == '$'){
                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 {
                            // printf("shift: %d\n", shift);
                            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, "*", &space);
                        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';
                    }
                    return result;
                } else {
                    name = end;
                    *end = ' ';

                }
            } else if (*name == '\\') {
                if (*(name + 1) == '*') {
                    cat(new, "*", &space);
                }
                name++;
                // 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++;
        }
        name++;

        //set the last char of orig to '\0', now name points to a string
        if (has_quote == true) {
            while (*end != '}') {
                if (*end == '\0') {
                    fprintf(stderr, "Error: missing '}'\n");
                    result = -1;
                    return result;
                }
                end++;
            }
            has_quote = !has_quote;
            *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;
}
```

```c
#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");
```

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

void exec_shift() {
    int cur_shift = 0;
    if (command[1] == NULL) {
        cur_shift = 1;
        shift += cur_shift;
    } else {
        cur_shift = atoi(command[1]);
        shift += cur_shift;
    }
    if ((args - shift) < 0) {
        fprintf(stderr, "can't shift that many arguments\n");
        // is_builtin = -1;
        // return is_builtin;
    } else {
        args = args - cur_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", asctime(localtime(&st.st_mtime))); //  print last modi
fied 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;
}
```

```
#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 2023-05-01 20:14:27-07:00 [TERM="xterm-256color" TTY="/dev/pts
/4" COLUMNS="130" LINES="28"]
^[[?2004h^[]0;zhengy@cf162-04: ~/csci347/csci347_s23/ush^G^[[01;32mzhengy@cf162-
04^[[00m:^[[01;34m~/csci347/csci347_s23/ush^[[00m$ cd ^[[7m/home/phil/public/csc
i347/testa2^[[27m^H^H^H^H^[[1P/home/phil/public/csci347/testa3
^[[?2004l
^[[?2004h^[]0;zhengy@cf162-04: /home/phil/public/csci347/testa3^G^[[01;32mzhengy
@cf162-04^[[00m:^[[01;34m/home/phil/public/csci347/testa3^[[00m$ ./try -H
^[[?2004l
mkdir: cannot create directory âM-^@M-^X/home/zhengy/347_test_a3âM-^@M-^Y: File
exists
~/347_test_a3 exists, use it anyway? (y/n) Y^H ^Hy
Cloning into 'csci347_s23'...
Username for 'https://gitlab.cs.wwu.edu': zhengy
Password for 'https://zhengy@gitlab.cs.wwu.edu':
remote: Enumerating objects: 336, done.^[[K
remote: Counting objects:   0% (1/312)^[[K
remote: Counting objects:   1% (4/312)^[[K
remote: Counting objects:   2% (7/312)^[[K
remote: Counting objects:   3% (10/312)^[[K
remote: Counting objects:   4% (13/312)^[[K
remote: Counting objects:   5% (16/312)^[[K
remote: Counting objects:   6% (19/312)^[[K
remote: Counting objects:   7% (22/312)^[[K
remote: Counting objects:   8% (25/312)^[[K
remote: Counting objects:   9% (29/312)^[[K
remote: Counting objects:  10% (32/312)^[[K
remote: Counting objects:  11% (35/312)^[[K
remote: Counting objects:  12% (38/312)^[[K
remote: Counting objects:  13% (41/312)^[[K
remote: Counting objects:  14% (44/312)^[[K
remote: Counting objects:  15% (47/312)^[[K
remote: Counting objects:  16% (50/312)^[[K
remote: Counting objects:  17% (54/312)^[[K
remote: Counting objects:  18% (57/312)^[[K
remote: Counting objects:  19% (60/312)^[[K
remote: Counting objects:  20% (63/312)^[[K
remote: Counting objects:  21% (66/312)^[[K
remote: Counting objects:  22% (69/312)^[[K
remote: Counting objects:  23% (72/312)^[[K
remote: Counting objects:  24% (75/312)^[[K
remote: Counting objects:  25% (78/312)^[[K
remote: Counting objects:  26% (82/312)^[[K
remote: Counting objects:  27% (85/312)^[[K
remote: Counting objects:  28% (88/312)^[[K
remote: Counting objects:  29% (91/312)^[[K
remote: Counting objects:  30% (94/312)^[[K
remote: Counting objects:  31% (97/312)^[[K
remote: Counting objects:  32% (100/312)^[[K
remote: Counting objects:  33% (103/312)^[[K
remote: Counting objects:  34% (107/312)^[[K
remote: Counting objects:  35% (110/312)^[[K
remote: Counting objects:  36% (113/312)^[[K
remote: Counting objects:  37% (116/312)^[[K
remote: Counting objects:  38% (119/312)^[[K
remote: Counting objects:  39% (122/312)^[[K
remote: Counting objects:  40% (125/312)^[[K
remote: Counting objects:  41% (128/312)^[[K
remote: Counting objects:  42% (132/312)^[[K
```

```
remote: Counting objects:  43% (135/312)^[[K
remote: Counting objects:  44% (138/312)^[[K
remote: Counting objects:  45% (141/312)^[[K
remote: Counting objects:  46% (144/312)^[[K
remote: Counting objects:  47% (147/312)^[[K
remote: Counting objects:  48% (150/312)^[[K
remote: Counting objects:  49% (153/312)^[[K
remote: Counting objects:  50% (156/312)^[[K
remote: Counting objects:  51% (160/312)^[[K
remote: Counting objects:  52% (163/312)^[[K
remote: Counting objects:  53% (166/312)^[[K
remote: Counting objects:  54% (169/312)^[[K
remote: Counting objects:  55% (172/312)^[[K
remote: Counting objects:  56% (175/312)^[[K
remote: Counting objects:  57% (178/312)^[[K
remote: Counting objects:  58% (181/312)^[[K
remote: Counting objects:  59% (185/312)^[[K
remote: Counting objects:  60% (188/312)^[[K
remote: Counting objects:  61% (191/312)^[[K
remote: Counting objects:  62% (194/312)^[[K
remote: Counting objects:  63% (197/312)^[[K
remote: Counting objects:  64% (200/312)^[[K
remote: Counting objects:  65% (203/312)^[[K
remote: Counting objects:  66% (206/312)^[[K
remote: Counting objects:  67% (210/312)^[[K
remote: Counting objects:  68% (213/312)^[[K
remote: Counting objects:  69% (216/312)^[[K
remote: Counting objects:  70% (219/312)^[[K
remote: Counting objects:  71% (222/312)^[[K
remote: Counting objects:  72% (225/312)^[[K
remote: Counting objects:  73% (228/312)^[[K
remote: Counting objects:  74% (231/312)^[[K
remote: Counting objects:  75% (234/312)^[[K
remote: Counting objects:  76% (238/312)^[[K
remote: Counting objects:  77% (241/312)^[[K
remote: Counting objects:  78% (244/312)^[[K
remote: Counting objects:  79% (247/312)^[[K
remote: Counting objects:  80% (250/312)^[[K
remote: Counting objects:  81% (253/312)^[[K
remote: Counting objects:  82% (256/312)^[[K
remote: Counting objects:  83% (259/312)^[[K
remote: Counting objects:  84% (263/312)^[[K
remote: Counting objects:  85% (266/312)^[[K
remote: Counting objects:  86% (269/312)^[[K
remote: Counting objects:  87% (272/312)^[[K
remote: Counting objects:  88% (275/312)^[[K
remote: Counting objects:  89% (278/312)^[[K
remote: Counting objects:  90% (281/312)^[[K
remote: Counting objects:  91% (284/312)^[[K
remote: Counting objects:  92% (288/312)^[[K
remote: Counting objects:  93% (291/312)^[[K
remote: Counting objects:  94% (294/312)^[[K
remote: Counting objects:  95% (297/312)^[[K
remote: Counting objects:  96% (300/312)^[[K
remote: Counting objects:  97% (303/312)^[[K
remote: Counting objects:  98% (306/312)^[[K
remote: Counting objects:  99% (309/312)^[[K
remote: Counting objects: 100% (312/312)^[[K
remote: Counting objects: 100% (312/312), done.^[[K
```

```
remote: Compressing objects:   0% (1/307)^[[K
remote: Compressing objects:   1% (4/307)^[[K
remote: Compressing objects:   2% (7/307)^[[K
remote: Compressing objects:   3% (10/307)^[[K
remote: Compressing objects:   4% (13/307)^[[K
remote: Compressing objects:   5% (16/307)^[[K
remote: Compressing objects:   6% (19/307)^[[K
remote: Compressing objects:   7% (22/307)^[[K
remote: Compressing objects:   8% (25/307)^[[K
remote: Compressing objects:   9% (28/307)^[[K
remote: Compressing objects:  10% (31/307)^[[K
remote: Compressing objects:  11% (34/307)^[[K
remote: Compressing objects:  12% (37/307)^[[K
remote: Compressing objects:  13% (40/307)^[[K
remote: Compressing objects:  14% (43/307)^[[K
remote: Compressing objects:  15% (47/307)^[[K
remote: Compressing objects:  16% (50/307)^[[K
remote: Compressing objects:  17% (53/307)^[[K
remote: Compressing objects:  18% (56/307)^[[K
remote: Compressing objects:  19% (59/307)^[[K
remote: Compressing objects:  20% (62/307)^[[K
remote: Compressing objects:  21% (65/307)^[[K
remote: Compressing objects:  22% (68/307)^[[K
remote: Compressing objects:  23% (71/307)^[[K
remote: Compressing objects:  24% (74/307)^[[K
remote: Compressing objects:  25% (77/307)^[[K
remote: Compressing objects:  26% (80/307)^[[K
remote: Compressing objects:  27% (83/307)^[[K
remote: Compressing objects:  28% (86/307)^[[K
remote: Compressing objects:  29% (90/307)^[[K
remote: Compressing objects:  30% (93/307)^[[K
remote: Compressing objects:  31% (96/307)^[[K
remote: Compressing objects:  32% (99/307)^[[K
remote: Compressing objects:  33% (102/307)^[[K
remote: Compressing objects:  34% (105/307)^[[K
remote: Compressing objects:  35% (108/307)^[[K
remote: Compressing objects:  36% (111/307)^[[K
remote: Compressing objects:  37% (114/307)^[[K
remote: Compressing objects:  38% (117/307)^[[K
remote: Compressing objects:  39% (120/307)^[[K
remote: Compressing objects:  40% (123/307)^[[K
remote: Compressing objects:  41% (126/307)^[[K
remote: Compressing objects:  42% (129/307)^[[K
remote: Compressing objects:  43% (133/307)^[[K
remote: Compressing objects:  44% (136/307)^[[K
remote: Compressing objects:  45% (139/307)^[[K
remote: Compressing objects:  46% (142/307)^[[K
remote: Compressing objects:  47% (145/307)^[[K
remote: Compressing objects:  48% (148/307)^[[K
remote: Compressing objects:  49% (151/307)^[[K
remote: Compressing objects:  50% (154/307)^[[K
remote: Compressing objects:  51% (157/307)^[[K
remote: Compressing objects:  52% (160/307)^[[K
remote: Compressing objects:  53% (163/307)^[[K
remote: Compressing objects:  54% (166/307)^[[K
remote: Compressing objects:  55% (169/307)^[[K
remote: Compressing objects:  56% (172/307)^[[K
remote: Compressing objects:  57% (175/307)^[[K
remote: Compressing objects:  58% (179/307)^[[K
```

```
remote: Compressing objects:  59% (182/307)^[[K
remote: Compressing objects:  60% (185/307)^[[K
remote: Compressing objects:  61% (188/307)^[[K
remote: Compressing objects:  62% (191/307)^[[K
remote: Compressing objects:  63% (194/307)^[[K
remote: Compressing objects:  64% (197/307)^[[K
remote: Compressing objects:  65% (200/307)^[[K
remote: Compressing objects:  66% (203/307)^[[K
remote: Compressing objects:  67% (206/307)^[[K
remote: Compressing objects:  68% (209/307)^[[K
remote: Compressing objects:  69% (212/307)^[[K
remote: Compressing objects:  70% (215/307)^[[K
remote: Compressing objects:  71% (218/307)^[[K
remote: Compressing objects:  72% (222/307)^[[K
remote: Compressing objects:  73% (225/307)^[[K
remote: Compressing objects:  74% (228/307)^[[K
remote: Compressing objects:  75% (231/307)^[[K
remote: Compressing objects:  76% (234/307)^[[K
remote: Compressing objects:  77% (237/307)^[[K
remote: Compressing objects:  78% (240/307)^[[K
remote: Compressing objects:  79% (243/307)^[[K
remote: Compressing objects:  80% (246/307)^[[K
remote: Compressing objects:  81% (249/307)^[[K
remote: Compressing objects:  82% (252/307)^[[K
remote: Compressing objects:  83% (255/307)^[[K
remote: Compressing objects:  84% (258/307)^[[K
remote: Compressing objects:  85% (261/307)^[[K
remote: Compressing objects:  86% (265/307)^[[K
remote: Compressing objects:  87% (268/307)^[[K
remote: Compressing objects:  88% (271/307)^[[K
remote: Compressing objects:  89% (274/307)^[[K
remote: Compressing objects:  90% (277/307)^[[K
remote: Compressing objects:  91% (280/307)^[[K
remote: Compressing objects:  92% (283/307)^[[K
remote: Compressing objects:  93% (286/307)^[[K
remote: Compressing objects:  94% (289/307)^[[K
remote: Compressing objects:  95% (292/307)^[[K
remote: Compressing objects:  96% (295/307)^[[K
remote: Compressing objects:  97% (298/307)^[[K
remote: Compressing objects:  98% (301/307)^[[K
remote: Compressing objects:  99% (304/307)^[[K
remote: Compressing objects: 100% (307/307)^[[K
remote: Compressing objects: 100% (307/307), done.^[[K
Receiving objects:   0% (1/336)
Receiving objects:   1% (4/336)
Receiving objects:   2% (7/336)
Receiving objects:   3% (11/336)
Receiving objects:   4% (14/336)
Receiving objects:   5% (17/336)
Receiving objects:   6% (21/336)
Receiving objects:   7% (24/336)
Receiving objects:   8% (27/336)
Receiving objects:   9% (31/336)
Receiving objects:  10% (34/336)
Receiving objects:  11% (37/336)
Receiving objects:  12% (41/336)
Receiving objects:  13% (44/336)
Receiving objects:  14% (48/336)
Receiving objects:  15% (51/336)
```

```
Receiving objects:  16% (54/336)
Receiving objects:  17% (58/336)
Receiving objects:  18% (61/336)
Receiving objects:  19% (64/336)
Receiving objects:  20% (68/336)
Receiving objects:  21% (71/336)
Receiving objects:  22% (74/336)
Receiving objects:  23% (78/336)
Receiving objects:  24% (81/336)
Receiving objects:  25% (84/336)
Receiving objects:  26% (88/336)
Receiving objects:  27% (91/336)
Receiving objects:  28% (95/336)
Receiving objects:  29% (98/336)
Receiving objects:  30% (101/336)
Receiving objects:  31% (105/336)
Receiving objects:  32% (108/336)
Receiving objects:  33% (111/336)
Receiving objects:  34% (115/336)
Receiving objects:  35% (118/336)
Receiving objects:  36% (121/336)
Receiving objects:  37% (125/336)
Receiving objects:  38% (128/336)
Receiving objects:  39% (132/336)
Receiving objects:  40% (135/336)
Receiving objects:  41% (138/336)
Receiving objects:  42% (142/336)
Receiving objects:  43% (145/336)
Receiving objects:  44% (148/336)
Receiving objects:  45% (152/336)
Receiving objects:  46% (155/336)
Receiving objects:  47% (158/336)
Receiving objects:  48% (162/336)
Receiving objects:  49% (165/336)
Receiving objects:  50% (168/336)
Receiving objects:  51% (172/336)
Receiving objects:  52% (175/336)
Receiving objects:  53% (179/336)
Receiving objects:  54% (182/336)
Receiving objects:  55% (185/336)
Receiving objects:  56% (189/336)
Receiving objects:  57% (192/336)
Receiving objects:  58% (195/336)
Receiving objects:  59% (199/336)
Receiving objects:  60% (202/336)
Receiving objects:  61% (205/336)
Receiving objects:  62% (209/336)
Receiving objects:  63% (212/336)
Receiving objects:  64% (216/336)
Receiving objects:  65% (219/336)
Receiving objects:  66% (222/336)
Receiving objects:  67% (226/336)
Receiving objects:  68% (229/336)
Receiving objects:  69% (232/336)
Receiving objects:  70% (236/336)
Receiving objects:  71% (239/336)
Receiving objects:  72% (242/336)
Receiving objects:  73% (246/336)
Receiving objects:  74% (249/336)
```

```
Receiving objects:  75% (252/336)
Receiving objects:  76% (256/336)
Receiving objects:  77% (259/336)
Receiving objects:  78% (263/336)
Receiving objects:  79% (266/336)
Receiving objects:  80% (269/336)
Receiving objects:  81% (273/336)
Receiving objects:  82% (276/336)
Receiving objects:  83% (279/336)
Receiving objects:  84% (283/336)
Receiving objects:  85% (286/336)
Receiving objects:  86% (289/336)
Receiving objects:  87% (293/336)
Receiving objects:  88% (296/336)
Receiving objects:  89% (300/336)
Receiving objects:  90% (303/336)
Receiving objects:  91% (306/336)
Receiving objects:  92% (310/336)
Receiving objects:  93% (313/336)
Receiving objects:  94% (316/336)
Receiving objects:  95% (320/336)
Receiving objects:  96% (323/336)
remote: Total 336 (delta 187), reused 0 (delta 0), pack-reused 24^[[K
Receiving objects:  97% (326/336)
Receiving objects:  98% (330/336)
Receiving objects:  99% (333/336)
Receiving objects: 100% (336/336)
Receiving objects: 100% (336/336), 361.85 KiB | 3.09 MiB/s, done.
Resolving deltas:   0% (0/193)
Resolving deltas:   1% (2/193)
Resolving deltas:   2% (4/193)
Resolving deltas:   3% (6/193)
Resolving deltas:   4% (8/193)
Resolving deltas:   5% (10/193)
Resolving deltas:   6% (12/193)
Resolving deltas:   7% (14/193)
Resolving deltas:   8% (16/193)
Resolving deltas:   9% (18/193)
Resolving deltas:  10% (20/193)
Resolving deltas:  11% (22/193)
Resolving deltas:  12% (24/193)
Resolving deltas:  13% (26/193)
Resolving deltas:  14% (28/193)
Resolving deltas:  15% (29/193)
Resolving deltas:  16% (31/193)
Resolving deltas:  17% (33/193)
Resolving deltas:  18% (35/193)
Resolving deltas:  19% (37/193)
Resolving deltas:  20% (39/193)
Resolving deltas:  21% (41/193)
Resolving deltas:  22% (43/193)
Resolving deltas:  23% (45/193)
Resolving deltas:  24% (47/193)
Resolving deltas:  25% (49/193)
Resolving deltas:  26% (51/193)
Resolving deltas:  27% (53/193)
Resolving deltas:  28% (55/193)
Resolving deltas:  29% (56/193)
Resolving deltas:  30% (58/193)
```

```
Resolving deltas:  31% (60/193)
Resolving deltas:  32% (62/193)
Resolving deltas:  33% (64/193)
Resolving deltas:  34% (66/193)
Resolving deltas:  35% (68/193)
Resolving deltas:  36% (70/193)
Resolving deltas:  37% (72/193)
Resolving deltas:  38% (74/193)
Resolving deltas:  39% (76/193)
Resolving deltas:  40% (78/193)
Resolving deltas:  41% (80/193)
Resolving deltas:  42% (82/193)
Resolving deltas:  43% (83/193)
Resolving deltas:  44% (85/193)
Resolving deltas:  45% (87/193)
Resolving deltas:  46% (89/193)
Resolving deltas:  47% (91/193)
Resolving deltas:  48% (93/193)
Resolving deltas:  49% (95/193)
Resolving deltas:  50% (97/193)
Resolving deltas:  51% (99/193)
Resolving deltas:  52% (101/193)
Resolving deltas:  53% (103/193)
Resolving deltas:  54% (105/193)
Resolving deltas:  55% (107/193)
Resolving deltas:  56% (109/193)
Resolving deltas:  57% (111/193)
Resolving deltas:  58% (112/193)
Resolving deltas:  59% (114/193)
Resolving deltas:  60% (116/193)
Resolving deltas:  61% (118/193)
Resolving deltas:  62% (120/193)
Resolving deltas:  63% (122/193)
Resolving deltas:  64% (124/193)
Resolving deltas:  65% (126/193)
Resolving deltas:  66% (128/193)
Resolving deltas:  67% (130/193)
Resolving deltas:  68% (132/193)
Resolving deltas:  69% (134/193)
Resolving deltas:  70% (136/193)
Resolving deltas:  71% (138/193)
Resolving deltas:  72% (139/193)
Resolving deltas:  73% (141/193)
Resolving deltas:  74% (143/193)
Resolving deltas:  75% (145/193)
Resolving deltas:  76% (147/193)
Resolving deltas:  77% (149/193)
Resolving deltas:  78% (151/193)
Resolving deltas:  79% (153/193)
Resolving deltas:  80% (155/193)
Resolving deltas:  81% (157/193)
Resolving deltas:  82% (159/193)
Resolving deltas:  83% (161/193)
Resolving deltas:  84% (163/193)
Resolving deltas:  85% (165/193)
Resolving deltas:  86% (166/193)
Resolving deltas:  87% (168/193)
Resolving deltas:  88% (170/193)
Resolving deltas:  89% (172/193)
```

```
Resolving deltas:  90% (174/193)
Resolving deltas:  91% (176/193)
Resolving deltas:  92% (178/193)
Resolving deltas:  93% (180/193)
Resolving deltas:  94% (182/193)
Resolving deltas:  95% (184/193)
Resolving deltas:  96% (186/193)
Resolving deltas:  97% (188/193)
Resolving deltas:  98% (190/193)
Resolving deltas:  99% (192/193)
Resolving deltas: 100% (193/193)
Resolving deltas: 100% (193/193), done.
Branch 'a3' set up to track remote branch 'a3' from 'origin'.
Switched to a new branch 'a3'
Making REF
gcc -g -Wall -c builtin.c
gcc -g -Wall -c strmode.c
gcc -g -Wall -o ush ush.o expand.o builtin.o strmode.o

run tests? y
Running ush

Script output same
Exit values correct

----- ERRS -----
% % % % % % % % % % % % % % % % %
----------------

Run error tests? y
Testing errors with /home/phil/public/csci347/testa3/ush.err and 5 arguments (sh
ould be 5)
Initial arguments are arg1, arg2, arg3, and arg4.
Doing a shift of 4 (no error)
We now have 1 argument, arg1 is ''.
Doing a shift of 4 and 1 (should be errors)
can't shift that many arguments
can't shift that many arguments
Now doing 'unshift 5' (should be an error)
Did an unshift ... number of args is 5, should be 5.
Testing sstat errors
End of error tests


Look at checked out files? y
^[[?2004h^[]0;zhengy@cf162-04: ~/347_test_a3/csci347_s23/ush^G^[[01;32mzhengy@cf
162-04^[[00m:^[[01;34m~/347_test_a3/csci347_s23/ush^[[00m$
^[[?2004l
^[[?2004h^[]0;zhengy@cf162-04: ~/347_test_a3/csci347_s23/ush^G^[[01;32mzhengy@cf
162-04^[[00m:^[[01;34m~/347_test_a3/csci347_s23/ush^[[00m$ exit
^[[?2004l
exit
Clean? y
cleaning
removing ~/347_test_a3
^[[?2004h^[]0;zhengy@cf162-04: /home/phil/public/csci347/testa3^G^[[01;32mzhengy
@cf162-04^[[00m:^[[01;34m/home/phil/public/csci347/testa3^[[00m$ exit
^[[?2004l
exit
```

```
Script done on 2023-05-01 20:15:18-07:00 [COMMAND_EXIT_CODE="0"]
```