Run

```
run:
cd the/myshell/path
make && ./myshell
```

Screenshots

```
 $ make && ./myshell
gcc -c utility.c
gcc -c myshell.c
myshell.c:46:36: warning: result of comparison against a string literal is
      unspecified (use strncmp instead) [-Wstring-compare]
    if (argc > 0 && argv[argc - 1] == "&") {
                                   ^  ~~~
1 warning generated.
gcc -o myshell utility.o myshell.o
Welcome to use Chiba's shell
----------------------------------------
/Users/Chiba/short-term-linux/hw4/myshell
$ dir
makefile        myshell.c       readme.md       utility.h
myshell         myshell.o       utility.c       utility.o
----------------------------------------
/Users/Chiba/short-term-linux/hw4/myshell
$ cd ../
----------------------------------------
/Users/Chiba/short-term-linux/hw4
$ cd myshell
----------------------------------------
/Users/Chiba/short-term-linux/hw4/myshell
$ environ
/bin/:/usr/bin/:
----------------------------------------
/Users/Chiba/short-term-linux/hw4/myshell
$ vadsfdsafas
Command vadsfdsafas not found.
```

```
----------------------------------------
/Users/Chiba/short-term-linux/hw4/myshell
$ cat myshell.c > tmp.c
----------------------------------------
/Users/Chiba/short-term-linux/hw4/myshell
$ cat tmp.c
/*******************************
* Author: Chiba(HUANG HUANG)  *
*******************************/

#include "utility.h"
//function declaration
int myExecu(list *);
void recurPipe(char *argv[], int);
char* rmSpace(char *);
//the main function
int main() {
  char *shellPath = "/myshell";
  int status;
  char command[105];
  pid_t pid2;
  head = NULL;
  puts("Welcome to use Chiba's shell");
  char *bspace = " ";
  int bgFlag;
  //preloaded path
  initPath();
  while(1) {
    bgFlag = 0;
```

```
 $ make && ./myshell
gcc -c utility.c
gcc -c myshell.c
myshell.c:46:36: warning: result of comparison against a string literal is
      unspecified (use strncmp instead) [-Wstring-compare]
    if (argc > 0 && argv[argc - 1] == "&") {
                                   ^  ~~~
1 warning generated.
```

```
gcc -o myshell utility.o myshell.o
Welcome to use Chiba's shell
------------------------------------------
/Users/Chiba/short-term-linux/hw4/myshell
$ dir
makefile        myshell.c       readme.md       utility.h
myshell         myshell.o       utility.c       utility.o
------------------------------------------
/Users/Chiba/short-term-linux/hw4/myshell
$ cd ../
------------------------------------------
/Users/Chiba/short-term-linux/hw4
$ cd myshell
------------------------------------------
/Users/Chiba/short-term-linux/hw4/myshell
$ environ
/bin/:/usr/bin/:
------------------------------------------
/Users/Chiba/short-term-linux/hw4/myshell
$ vadsfdsafas
Command vadsfdsafas not found.
------------------------------------------
/Users/Chiba/short-term-linux/hw4/myshell
$ cat myshell.c > tmp.c
------------------------------------------
/Users/Chiba/short-term-linux/hw4/myshell
$ cat tmp.c
/*******************************
* Author: Chiba(HUANG HUANG)  *
*******************************/

#include "utility.h"
//function declaration
int myExecu(list *);
void recurPipe(char *argv[], int);
char* rmSpace(char *);
//the main function
int main() {
  char *shellPath = "/myshell";
```

```
  int status;
  char command[105];
  pid_t pid2;
  head = NULL;
  puts("Welcome to use Chiba's shell");
  char *bspace = " ";
  int bgFlag;
  //preloaded path
  initPath();
  while(1) {
    bgFlag = 0;
```

Code

```
$ make && ./myshell
gcc -c utility.c
gcc -c myshell.c
myshell.c:46:36: warning: result of comparison against a string literal is
      unspecified (use strncmp instead) [-Wstring-compare]
    if (argc > 0 && argv[argc - 1] == "&") {
                                   ^  ~~~
1 warning generated.
gcc -o myshell utility.o myshell.o
Welcome to use Chiba's shell
-------------------------------------------
/Users/Chiba/short-term-linux/hw4/myshell
$ dir
makefile    myshell.c   readme.md   utility.h
myshell     myshell.o   utility.c   utility.o
-------------------------------------------
/Users/Chiba/short-term-linux/hw4/myshell
$ cd ../
-------------------------------------------
/Users/Chiba/short-term-linux/hw4
$ cd myshell
-------------------------------------------
/Users/Chiba/short-term-linux/hw4/myshell
$ environ
/bin/:/usr/bin/:
-------------------------------------------
/Users/Chiba/short-term-linux/hw4/myshell
$ vadsfdsafas
Command vadsfdsafas not found.
-------------------------------------------
/Users/Chiba/short-term-linux/hw4/myshell
$ cat myshell.c > tmp.c
```

```
$ cat myshell.c tmp.c
-------------------------------------------
/Users/Chiba/short-term-linux/hw4/myshell
$ cat tmp.c
/*****************************
* Author: Chiba(HUANG HUANG)  *
*****************************/

#include "utility.h"
//function declaration
int myExecu(list *);
void recurPipe(char *argv[], int);
char* rmSpace(char *);
//the main function
int main() {
  char *shellPath = "/myshell";
  int status;
  char command[105];
  pid_t pid2;
  head = NULL;
  puts("Welcome to use Chiba's shell");
  char *bspace = " ";
  int bgFlag;
  //preloaded path
  initPath();
  while(1) {
    bgFlag = 0;
    printf("-------------------------------------------\n");
    getCurrentPath();
    printf("$ ");
    argHead = NULL;
    fflush(stdin);
    char* cmdBegin;
    char cmd2[4096];
    fgets(cmd2,4096,stdin);
    //
    char* cmd = rmSpace(cmd2);
    int argc = countSpace(cmd)+1;
    char *argv[argc];
    int space;
    space = strcspn(cmd,bspace);
    //readin all the arguments
    char* arg  =  strtok(cmd, bspace);
    int count = 0;
    while (arg){
      argv[count] = arg;
      arg = strtok(NULL, bspace);
      count++;
    }
    if (argc > 0 && argv[argc - 1] == "&") {
      bgFlag = 1;
      argc--;
    }
    count = argc;
```

```
    count = argc;
    returnMod(argv[count - 1]);
    argv[count] = (char*)0;
    if (strcmp(argv[0], "quit") == 0) {
        puts("Quit..");
        exit(-1); }
    if (strcmp(argv[0], "environ") == 0) {
      if (count == 1) printPath();
      continue;
    }
    if (strcmp(argv[0], "cd") == 0){
      if (argc == 1) {
        getCurrentPath();
      } else if (chdir(argv[1]) < 0){
        fprintf(stderr, "Error: %s\n", strerror(errno));
      };
      continue;
    }
    if (strcmp(argv[0], "clr") == 0) {
      clear();
      continue;
    }
    if (strcmp(argv[0], "dir") == 0) {
      argv[0] = dir();
    }
    if (strcmp(argv[0], "help") == 0) {
      help();
      if (bgFlag == 0) {
        wait(&status);
      }
      continue;
    }
    if ((pid2 = fork()) < 0){
      fprintf(stderr, "Error: %s\n", strerror(errno));
      return -1;
    }
    if (pid2 == 0) {
      recurPipe(argv,count);
      exit(0);
    } else {
      wait(&status);
    }
  }
}

void recurPipe(char *argv[],int count){
  int status;
  pid_t pid;
  int pipeCnt = 0;
  int i, j, k;
  int count2;
  for(i = 0; i < count; i++) {
    if(strcmp(argv[i], "|") == 0) {
      pipeCnt++;
```

```
    pipeCnt++;
  }
}
int total[pipeCnt + 2];
total[0] = 0;
total[pipeCnt + 1] = count;
int l = 0, m = 1;
for(; l < count; l++) {
  if(strcmp(argv[l], "|") == 0) {
    total[m] = l + 1;
    m++;
  }
}
char **addr;
char **newaddr;
int o = 0;
int n = 0;
if(pipeCnt == 0){
  newaddr = &argv[total[n]];
  o = total[n + 1] - total[n];
  addr = shortArray(newaddr,o);
  addPath2(argv[total[n]], addr);
} else {
  for(; n < pipeCnt + 1; n++){
    newaddr = &argv[total[n]];
    o = total[n + 1] - total[n] - 1;
    if(n == pipeCnt) o++;
    addr = shortArray(newaddr, o);
    addPath2(argv[total[n]], addr);
  }
}
//the following part with reference to :
//http://stackoverflow.com/questions/8389033/implementation-of-multiple-pipes-in-c
int p = 0;
int numPipes = pipeCnt;
int pipefds[2 * numPipes];
for(i = 0; i < (numPipes); i++){
  if(pipe(pipefds + i * 2) < 0) {
    perror("couldn't pipe");
    exit(1);
  }
}
while(argHead){
  pid = fork();
  if(pid == 0){
    if(argHead->next){
      if(dup2(pipefds[p * 2 + 1], 1) < 0) exit(1);
    }
    if(p != 0){
      if(dup2(pipefds[(p - 1) * 2], 0) < 0){
        perror(" dup2");
        exit(1);
      }
```

```
      }
      for (i = 0; i < 2 * numPipes; i++) close(pipefds[i]);
      if(myExecu(argHead) < 0) {
        fprintf(stderr, "Error: %s\n", strerror(errno));
        perror(argHead->path);
        exit(1);
      }
    } else if (pid < 0) {
      perror("error");
      exit(1);
    }
    argHead = argHead->next;
    p++;
  }
  for(i  =  0; i < 2 * numPipes; i++) close(pipefds[i]);
  for(i  =  0; i < numPipes + 1; i++) wait(&status);
}
//
char* rmSpace(char *raw) {
    int tmp = 0;
    while(raw[tmp] == ' ') tmp++;
    return &raw[tmp];
}
//function to handle each pipeline
int myExecu(list* top) {
  list* it = top;
  char **ars = top->arguments;
  int count = 0;
  while(ars[count] != (char*)0){
    count++;
  }
  int breakpoint = count;
  int tcounter = 0;
  while(ars[tcounter]!= (char*)0) {
    if(strcmp(ars[tcounter],">") == 0 ||strcmp(ars[tcounter],"<") == 0) {
      breakpoint = tcounter;
      break;
    }
    tcounter++;
  }

  int fileopen;
  int fileread;

  char * argv2[breakpoint+1];
  for(tcounter = 0;tcounter<breakpoint;tcounter++){
    argv2[tcounter] = ars[tcounter];
  }
  argv2[breakpoint] = (char*)0;

  for(tcounter = 0;tcounter<count;tcounter++){
    if(strcmp(ars[tcounter],">") == 0) {
      tcounter++;
```

```
      if((fileopen = open(ars[tcounter],(O_CREAT|O_RDWR),0644))<0) {
        fprintf(stderr, "Error: %s\n", strerror(errno));
        exit(-1);
      } else {
        if(dup2(fileopen,1) < 0) {
          fprintf(stderr, "Error: %s\n", strerror(errno));
          exit(-1);
        }
      }
    } else if(strcmp(ars[tcounter],"<") == 0) {
      tcounter++;
      if((fileopen = open(ars[tcounter],O_RDONLY))<0){
        fprintf(stderr, "Error: %s\n", strerror(errno));
        exit(-1);
      } else {
        if(dup2(fileopen,0) < 0) {
          fprintf(stderr, "Error: fail to redirect\n");
          exit(-1);
        }
      }
    }
  }
  //system provided functions
  int t = 0;
  if(head == NULL){
    fprintf(stderr, "Error: No path to execute command\n");
    exit(0);
  }
  list* iter = head;
  char* command = iter->path;
  strcat(command, top->path);
  while((t = execv(command, argv2) == -1) && iter!= NULL) {
    iter = iter->next;
    command = iter->path;
    if(command == NULL) {
      fprintf(stderr, "Command %s not found.\n", ars[0]);
      exit(0);
    }
    strcat(command,ars[0]);
  }
  close(fileopen);
  exit(0);
}
-------------------------------------------
/Users/Chiba/short-term-linux/hw4/myshell
$ quit
Quit..
```