Assignment3 Report

姓名：Yitong WANG(王奕童) 11910104@mail.sustech.edu.cn

学号：11910104

实验课时段：周五5-6节

实验课教师：Yun SHEN(沈昀) sheny@mail.sustech.edu.cn

实验课SA:

- Yining TANG(汤怡宁) 11811237@mail.sustech.edu.cn
- Yushan WANG(王宇杉) 11813002@mail.sustech.edu.cn

Q1 `Three Easy Pieces`

Q1-1 Three Easy Pieces Explanation:

- Virtualizing（虚拟化）

参考书中第2页对于虚拟化的定义：

> The primary way the OS does this is through a general technique that we call **virtualization**. That is, the OS takes a **physical** resource (such as the processor, or memory, or a disk) and transforms it into a more general, powerful, and easy-to-use **virtual** form of itself. Thus, we sometimes refer to the operating system as a **virtual machine**.

虚拟化是指操作系统将硬件设备的物理资源进行一层抽象与转换，转换成一个更加普适，易用和强大的虚拟形式，从而实现底层硬件与上层应用的隔离，将对操作系统资源的访问提供统一抽象化的接口。

- Concurrency（并发度）

参考书中第7页对于并发度的说明：

Another main theme of this book is **concurrency**. We use this conceptual term to refer to a host of problems that arise, and must be addressed, when working on many things at once (i.e., concurrently) in the same program. The problems of concurrency arose first within the operating system itself; as you can see in the examples above on virtualization, the OS is juggling many things at once, first running one process, then another, and so forth. As it turns out, doing so leads to some deep and interesting problems.

并发度是指在操作系统同时处理多个进程的运行，但是对于任一时刻，只能有一个进程在执行，多个进程之间会相互切换着运行。

- Persistence（持久度）

书中对于持久度的说明如下：

The third major theme of the course is **persistence**. In system memory, data can be easily lost, as devices such as DRAM store values in a **volatile** manner; when power goes away or the system crashes, any data in memory is lost. Thus, we need hardware and software to be able to store data **persistently**; such storage is thus critical to any system as users care a great deal about their data.

持久度就是说操作系统的硬件和软件对于数据持久保存的能力，具体的实现有文件系统等等。

Q1-2 Three Easy Pieces Mapping

- Virtualizing: Chapter 3-5
- Concurrency: Chapter 6-8
- Persistence: Chapter 12-15

Q2 Context Switch

先参考书中第9页对上下文切换的说明：

If the decision is made to switch, the OS then executes a low-level piece of code which we refer to as a **context switch**. A context switch is conceptually simple: all the OS has to do is save a few register values for the currently-executing process (onto its kernel stack, for example) and restore a few for the soon-to-be-executing process (from its kernel stack). By doing so, the OS thus ensures that when the return-from-trap instruction is finally executed, instead of returning to the process that was running, the system resumes execution of another process.

因此上下文切换主要有以下几个步骤：

- 将前一个任务的CPU上下文（CPU寄存器和程序计数器）保存
- 加载新任务的上下文的CPU寄存器和程序计数器
- 跳转程序计数器所指向的新位置
- 运行新的任务、

Q3 `fork()` and `exit()`

Q3-1 `fork()`

- 系统调用机制：
  - `fork()` 的系统调用将会创建一个与父进程几乎完全一样的新子进程。其中，当前进程确定自己是父进程/子进程的方法是根据 `fork()` 的返回值：如果是0则为子进程；否则，则是父进程环境，返回值为新创建子进程的进程ID。
  - `fork()` 系统调用创建的新进程，内存布局和数据几乎完全相同。其中他们会在只读存储区共享相同的物理内存页；其余可读可写的数据段，堆栈内存等信息，每个进程是独立创建的。
- 地址空间：
  - 地址空间分布：分为四段：程序段（Code Segment），数据段（Data Segment），栈空间（Stack）和堆空间（Heap）
  - fork系统调用后，父进程和子进程共享程序段（因为该部分是共享只读的），其余部分都是独立复制出来的（因为都是可读可写，需要相互隔离）
- PCB:
  - PCB是进程控制块，是对程序运行的动态描述
  - `fork()` 调用的时候，父进程会将PCB信息拷贝给子进程
- CPU调度器：
  - 在 `fork()` 创建了子进程后，CPU调度器会决定下一步哪个进程先运行。
- 上下文切换
  - 当 `fork()` 系统调用的时候，操作系统需要进入内核态。这时候就需要进行进程的上下文切换，需要保存PCB，程序段和数据信息。

- 返回值
  - 如返回结果是-1，则说明生成子进程失败，一般原因是进程号全部被占用了
  - 如调用fork的是父进程，则返回生成的子进程的进程id
  - 如调用fork的是子进程，则返回值为0

Q3-2 `exit()`

`exit()` 主要做了以下几个流程：

- 内核释放所有该进程占用的内存
- 关闭所有该进程打开的文件列表
- 释放该进程相关的所有用户空间的内存

关于“僵尸态”：

- 该进程仍然在内核的进程表中，需要等待父进程收集它的退出码（因此该进程就处于僵尸态）
- 内核将子进程的退出码 `SIGCHLD` 通知父进程，等待父进程处理

关于与 `wait()` 的联系：

- 父进程没有处于 `wait()` 状态，则子进程通知的 `SIGCHLD` 无效，父进程不会处理；
- 父进程处于 `wait()` 状态，则内核会注册一个子进程的信号处理流程。
  - 处理流程是首先接收并移除 `SIGCHLD` 信号，其次再内核空间中移除子进程，从而解除“僵尸态
- 内核解除注册信号处理流程，并且将终止掉的子进程的pid作为wait()的返回值。

Q4 Transferring from user to OS

Q4-1 Three Methods

先参考课件上对于这部分内容的说明：

- Once a process runs on a CPU, it only gives back the control of a CPU
  - when it makes a system call
  - when it raises an exception
  - when an interrupt occurs

因此CPU的控制由用户态到内核态的转换有三种可能：

- 系统调用
- 异常
- 中断

Q4-2 Comparison

从以下几个方面比较：

概念

- 系统调用：应用程序主动向OS发出服务请求
- 异常：非法指令序列，或者非法内存访问等等原因导致的指令执行失败
- 中断：硬件设备发出的处理请求

产生源头

- 系统调用：应用程序请求操作系统完成相关任务，比如说fork，wait或者exec
- 异常：应用程序预料之外的行为，比如说非法的地址访问
- 中断：外部设备引起，比如说时钟中断，或者是I/O中断

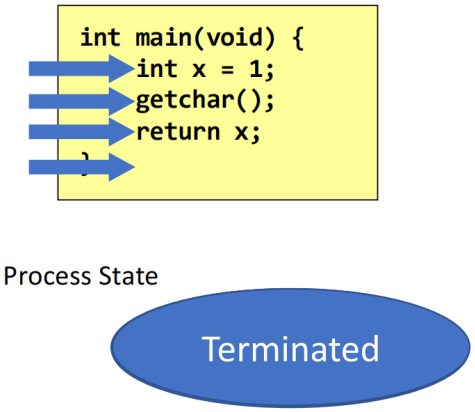
处理机制

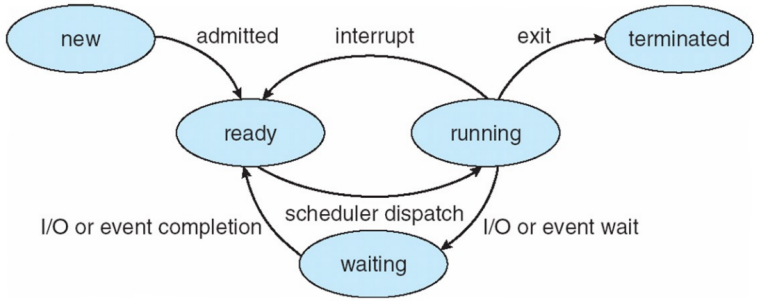
- 系统调用：等待操作系统响应
- 异常：杀死当前进程，或者是重新执行产生意料之外结果的指令
- 中断：持续，等待中断处理

Q5 Process Life Cycle

进程生命周期首先参考slide上这张图：

Process Life Cycle





- new: 分配了PCB，但并无进程所需资源，创建工作不完整，进程不能调度运行
- ready: 进程已分配到除CPU以外的所有资源，处于CPU的等待态
- running: 进程获得CPU，正在执行
- waiting: 进程由于一些事件，不能继续执行，会释放CPU占用并进入阻塞状态
- terminated: 进程自然结束，或者异常终止。进程不再能够执行，但是仍然在系统进程表中有对应的记录，供其他进程收集。

Reason Description

- new -> ready：进程正在被创建，OS正在为其分配所需要的资源
- ready -> running：进程通过CPU调度器获得了CPU的相关资源
- running -> ready: 进程被调度，有可能是被回收了CPU资源，也有可能是系统中断
- running -> waiting：系统调用，一般是主动请求等待事件发生，或者说申请系统资源
- waiting -> ready：系统资源申请成功，或者等待请求的事件发生了
- running -> terminated：进程自然结束，或者出现了不可预料的异常使得异常终止

Q6 `myshell` implementation

Code Screenshots:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <pwd.h>
#include <ctype.h>
#include <sys/wait.h>
#include <sys/stat.h>
#include "memory.h"

#define boolean int
#define TRUE 1
#define FALSE 0
#define INPUT_BUFFERSIZE 1024
#define usernameLength 64
#define hostnameLength 64
#define workSpaceLength 128
#define msleep(x) usleep(x*1000)

char *username = NULL;
char *hostname = NULL;
char *currentWorkSpace = NULL;
char *outputWorkSpace = NULL;

char *inputCommand = NULL;

char *initEmptyString(int length) {
    char *str = malloc( size: sizeof(char) * length);
    memset( s: str,  c: 0,  n: sizeof(char) * length);
    return str;
}

void getusername() {
    struct passwd *pwd = getpwuid( uid: getuid());
    strcpy( dest: username,  src: pwd->pw_name);
}

boolean isSameUser() {
    int length = (int) strlen( s: username);

    boolean same = TRUE;
    char *expectedFirst = "/home/";
    for (int i = 0; i < 6; ++i) {
        if (tolower( c: currentWorkSpace[i]) != tolower( c: expectedFirst[i])) {
            same = FALSE;
            break;
        }
    }
```

```
    if (!same) {
        return FALSE;
    }

    for (int i = 6; i < 6 + length && i < strlen( s: currentWorkSpace); ++i) {
        if (tolower( c: username[i - 6]) != tolower( c: currentWorkSpace[i])) {
            same = FALSE;
            break;
        }
    }

    return same;
}

int readLine(char *buff, int bufferSize) {
    memset( s: buff,  c: 0,  n: sizeof(char) * bufferSize);
    int c = 0;
    int length = 0;

    while ((c = getchar()) != EOF && length < bufferSize) {

        if (c == '\n') {
            break;
        }
        buff[length++] = (char) c;
    }

    buff[length] = '\0';
    return length;
}

void setOutputWorkSpace() {
    if (isSameUser()) {
        outputWorkSpace[0] = '~';
        int delta = (int) strlen( s: username) + 6;
        strcpy( dest: outputWorkSpace + 1,  src: currentWorkSpace + delta);
    } else {
        strcpy( dest: outputWorkSpace,  src: currentWorkSpace);
    }
}
```

```
boolean checkInputLine(char *inputLine) {
    char *validCommands[] = { [0]: "cd",  [1]: "ps",  [2]: "ls",
                              [3]: "pwd",  [4]: "exit",  [5]: "touch",
                              [6]: "mkdir",  [7]: "rm"};

    int firstBlankIndex = -1;

    boolean nonBlank = FALSE;
    int nonBlankStart = 0;
    for (int i = 0; i < strlen( s: inputLine); ++i) {
        if (inputLine[i] == ' ') {
            if (nonBlank) {
                firstBlankIndex = i;
                break;
            }
        } else {
            if (!nonBlank) {
                nonBlank = TRUE;
                nonBlankStart = i;
            }

        }
    }

    if (!nonBlank) {
        return FALSE;
    }

    if (firstBlankIndex == -1) {
        firstBlankIndex = strlen( s: inputLine);
        nonBlankStart = 0;
    }

    boolean fit = FALSE;
    for (int i = 0; i < sizeof(validCommands) / sizeof(char *); ++i) {
        if (strlen( s: validCommands[i]) != firstBlankIndex - nonBlankStart) {
            continue;
        }

        boolean tempFit = TRUE;
        for (int j = nonBlankStart; j < firstBlankIndex; ++j) {
            if (validCommands[i][j - nonBlankStart] != inputLine[j]) {
                tempFit = FALSE;
                break;
            }
        }

        if (tempFit) {
```

```
            fit = TRUE;
            break;
        }
    }

    if (!fit) {
        memset( s: inputCommand,  c: 0,  n: sizeof(char) * strlen( s: inputCommand));
        memcpy( dest: inputCommand,  src: inputLine + nonBlankStart,
                n: sizeof(char) * (firstBlankIndex - nonBlankStart));
    }

    return fit;
}

```

```c
boolean judgeCD(char *inputLine) {
    if (strncasecmp( s1: inputLine,  s2: "cd",  n: 2) == 0) {
        char *restPath = initEmptyString( length: INPUT_BUFFERSIZE);
        strcpy( dest: restPath,  src: inputLine + strlen( s: "cd") + 1);

        if (strlen( s: restPath) == 0 ||
        (strlen( s: restPath) == 1 && restPath[0] == '~')) {
            chdir( path: getenv( name: "HOME"));
        } else {
            if (chdir( path: restPath) != 0) {
                printf( format: "cd: %s: No such file or directory!\n", restPath);
            }

        }

        free( ptr: restPath);
        restPath = NULL;
        return TRUE;
    }

    return FALSE;
}

boolean judgeTouch(char *inputLine) {
    if (strncasecmp( s1: inputLine,  s2: "touch",  n: 5) == 0) {
        char *restPath = initEmptyString( length: INPUT_BUFFERSIZE);
        strcpy( dest: restPath,  src: inputLine + strlen( s: "touch") + 1);

        if (strlen( s: restPath) == 0) {
            printf( format: "Missing File Name Parameter!\n");
        } else {

            FILE *fileP = fopen( filename: restPath,  modes: "r");
            if (fileP == NULL) {
                fileP = fopen( filename: restPath,  modes: "w");
                if (fileP == NULL) {
                    printf( format: "touch: %s: Cannot create file!\n", restPath);
                }
            }

            fclose( stream: fileP);

            fileP = NULL;


        }
        free( ptr: restPath);
        restPath = NULL;
        return TRUE;
```

```
    }

    return FALSE;
}

```

```c
boolean judgeMkdir(char *inputLine) {
    if (strncasecmp( s1: inputLine,  s2: "mkdir",  n: 5) == 0) {
        char *restPath = initEmptyString( length: INPUT_BUFFERSIZE);
        strcpy( dest: restPath,  src: inputLine + strlen( s: "mkdir") + 1);

        if (strlen( s: restPath) == 0) {
            printf( format: "Missing Directory Name Parameter!\n");
        } else {

            if (mkdir( path: restPath,  mode: S_IRWXU | S_IRGRP |
            S_IWGRP | S_IROTH) == 0) {
                printf( format: "mkdir: %s: Directory is created successfully.\n"
                        , restPath);
            } else {
                printf( format: "create file error!\n");
            }

        }
        free( ptr: restPath);
        restPath = NULL;
        return TRUE;
    }

    return FALSE;
}

boolean judgeRM(char *inputLine) {
    if (strncasecmp( s1: inputLine,  s2: "rm",  n: 2) == 0) {
        char *restPath = initEmptyString( length: INPUT_BUFFERSIZE);
        strcpy( dest: restPath,  src: inputLine + strlen( s: "rm") + 1);

        if (strlen( s: restPath) == 0) {
            printf( format: "Missing File Name Parameter!\n");
        } else {

            if (remove( filename: restPath) == 0) {
                printf( format: "rm successfully.\n");
            } else {
                printf( format: "rm error!\n");
            }

        }
        free( ptr: restPath);
        restPath = NULL;
        return TRUE;
    }

    return FALSE;
}
```

```c
int main() {

    username = initEmptyString( length: usernameLength);
    hostname = initEmptyString( length: hostnameLength);
    currentWorkSpace = initEmptyString( length: workSpaceLength);
    outputWorkSpace = initEmptyString( length: workSpaceLength);
    inputCommand = initEmptyString( length: workSpaceLength);

    getusername();
    gethostname( name: hostname,  len: hostnameLength);


    char *inputLine = initEmptyString( length: INPUT_BUFFERSIZE);

    while (TRUE) {
        getcwd( buf: currentWorkSpace,  size: workSpaceLength);
        setOutputWorkSpace();
        printf( format: "%s@%s:%s ", username, hostname, outputWorkSpace);
        readLine( buff: inputLine,  bufferSize: INPUT_BUFFERSIZE);

        if (!checkInputLine(inputLine)) {
            printf( format: "No such command: %s\n", inputCommand);
            continue;
        }

        if (strncasecmp( s1: inputLine,  s2: "exit",  n: 4) == 0) {
            break;
        }

        if (judgeCD(inputLine)) {
            continue;
        }

        if (judgeTouch(inputLine)) {
            continue;
        }

        if (judgeMkdir(inputLine)) {
            continue;
        }

        if (judgeRM(inputLine)) {
            continue;
        }

        char *strs[] = { [0]: "ps",  [1]: "ls",  [2]: "pwd"};

        for (int i = 0; i < sizeof(strs) / sizeof(char *); ++i) {
            if (strncasecmp( s1: inputLine,  s2: strs[i],  n: strlen( s: strs[i])) == 0)

```

```c
                char *temp = initEmptyString( length: INPUT_BUFFERSIZE);

                strcpy( dest: temp,  src: inputLine + strlen( s: strs[i]) + 1);

                printf( format: "%s", temp);

                if (fork() == 0) {

                    if (strlen( s: temp) == 0) {
                        if (strs[i][0] == 'l' && strs[i][1] == 's') {
                            execlp( file: strs[i],  arg: strs[i], currentWorkSpace, NUL
                        } else {
                            execlp( file: strs[i],  arg: strs[i], NULL);
                        }

                    } else {
                        execlp( file: strs[i],  arg: strs[i], temp, NULL);
                    }

                } else {
                    wait( stat_loc: NULL);
```

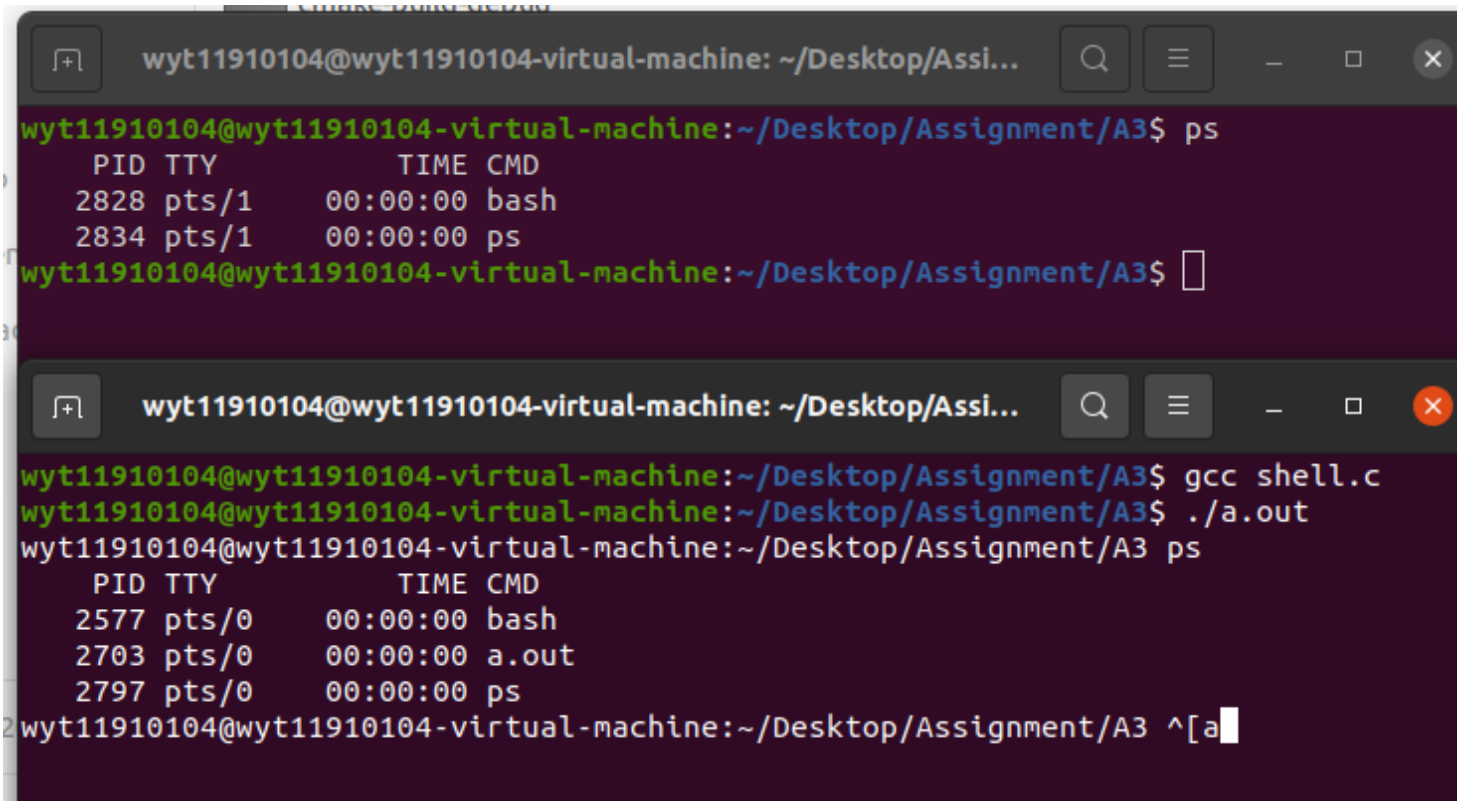
Code screenshots
Image was copied to the clipboard

```c
                }

                free( ptr: temp);
                temp = NULL;
            }
        }


    }

    free( ptr: username);
    username = NULL;
    free( ptr: hostname);
    hostname = NULL;
    free( ptr: currentWorkSpace);
    currentWorkSpace = NULL;
    free( ptr: outputWorkSpace);
    outputWorkSpace = NULL;
    free( ptr: inputCommand);
    inputCommand = NULL;
    free( ptr: inputLine);
    inputLine = NULL;

    return EXIT_SUCCESS;
}
```

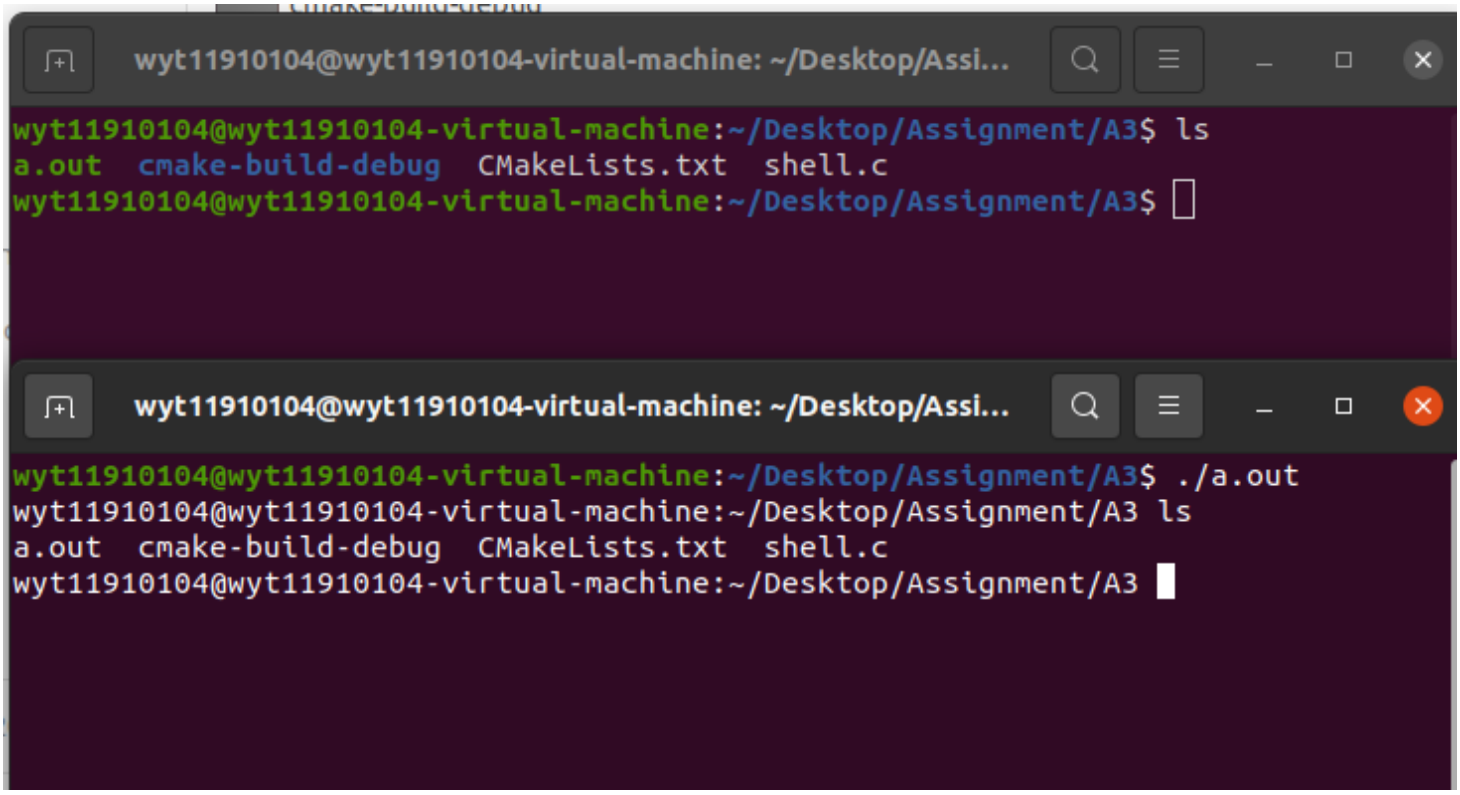
Function Screenshots:

Basic Functions:

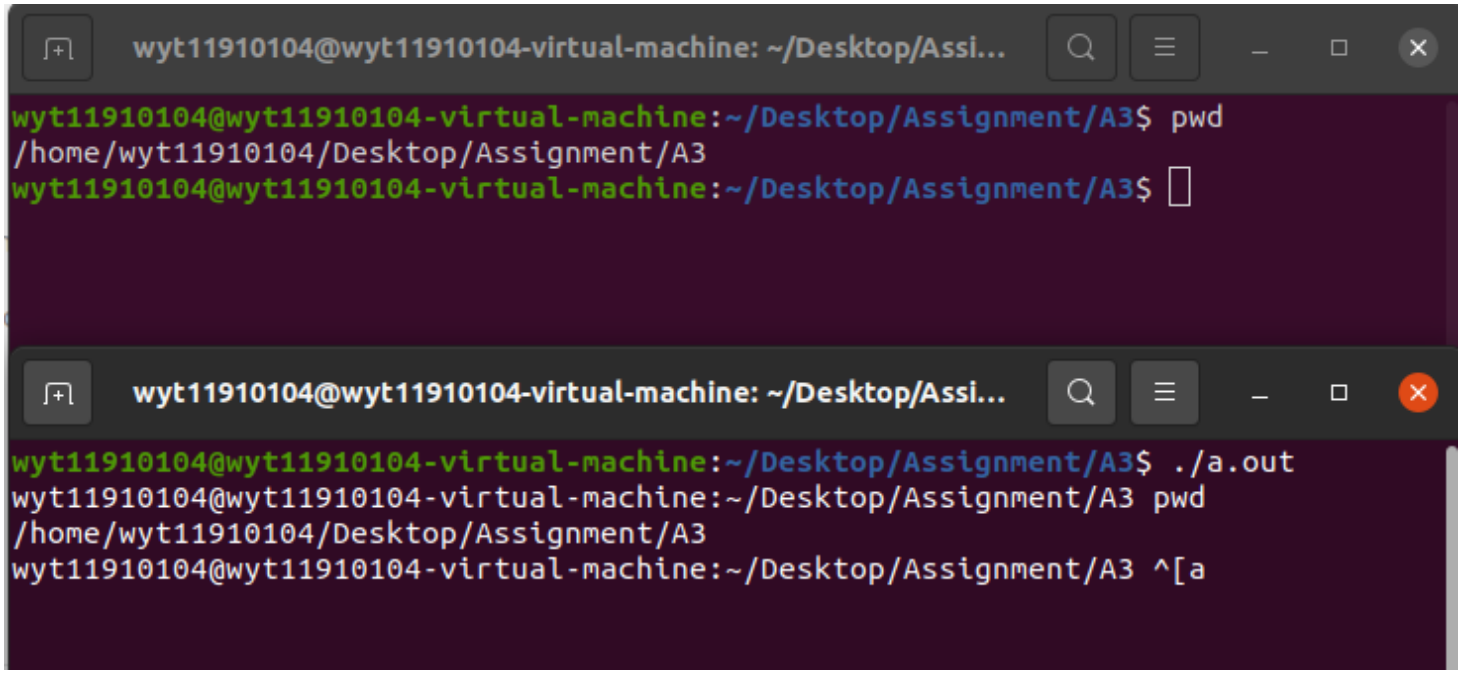
- ps



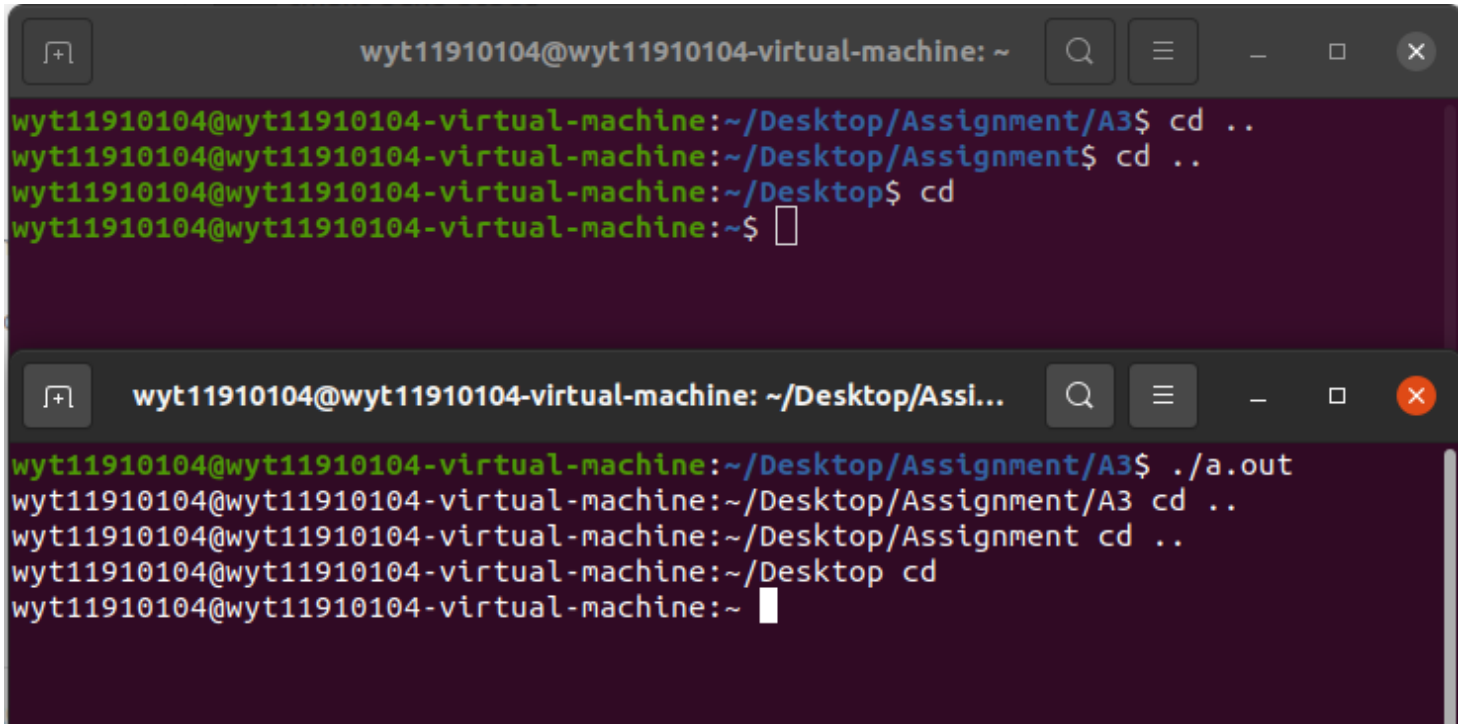
- ls



- pwd


wyt11910104@wyt11910104-virtual-machine: ~/Desktop/Assi...
wyt11910104@wyt11910104-virtual-machine:~/Desktop/Assignment/A3$ pwd
/home/wyt11910104/Desktop/Assignment/A3
wyt11910104@wyt11910104-virtual-machine:~/Desktop/Assignment/A3$
wyt11910104@wyt11910104-virtual-machine: ~/Desktop/Assi...
wyt11910104@wyt11910104-virtual-machine:~/Desktop/Assignment/A3$ ./a.out
wyt11910104@wyt11910104-virtual-machine:~/Desktop/Assignment/A3 pwd
/home/wyt11910104/Desktop/Assignment/A3
wyt11910104@wyt11910104-virtual-machine:~/Desktop/Assignment/A3 ^[a

- cd


wyt11910104@wyt11910104-virtual-machine: ~
wyt11910104@wyt11910104-virtual-machine:~/Desktop/Assignment/A3$ cd ..
wyt11910104@wyt11910104-virtual-machine:~/Desktop/Assignment$ cd ..
wyt11910104@wyt11910104-virtual-machine:~/Desktop$ cd
wyt11910104@wyt11910104-virtual-machine:~$
wyt11910104@wyt11910104-virtual-machine: ~/Desktop/Assi...
wyt11910104@wyt11910104-virtual-machine:~/Desktop/Assignment/A3$ ./a.out
wyt11910104@wyt11910104-virtual-machine:~/Desktop/Assignment/A3 cd ..
wyt11910104@wyt11910104-virtual-machine:~/Desktop/Assignment cd ..
wyt11910104@wyt11910104-virtual-machine:~/Desktop cd
wyt11910104@wyt11910104-virtual-machine:~

- exit

```
wyt11910104@wyt11910104-virtual-machine: ~/Desktop/Assi...
wyt11910104@wyt11910104-virtual-machine:~/Desktop/Assignment/A3$ ./a.out
wyt11910104@wyt11910104-virtual-machine:~/Desktop/Assignment/A3 cd ..
wyt11910104@wyt11910104-virtual-machine:~/Desktop/Assignment cd ..
wyt11910104@wyt11910104-virtual-machine:~/Desktop cd
wyt11910104@wyt11910104-virtual-machine:~ exit
wyt11910104@wyt11910104-virtual-machine:~/Desktop/Assignment/A3$
```

Bonus Functions:

- 检查输入的指令是否合法:

```
wyt11910104@wyt11910104-virtual-machine: ~/Desktop/Assi...
wyt11910104@wyt11910104-virtual-machine:~/Desktop/Assignment/A3$ gcc shell.c
wyt11910104@wyt11910104-virtual-machine:~/Desktop/Assignment/A3$ ./a.out
wyt11910104@wyt11910104-virtual-machine:~/Desktop/Assignment/A3 asdf
No such command: asdf
wyt11910104@wyt11910104-virtual-machine:~/Desktop/Assignment/A3 cd ..
wyt11910104@wyt11910104-virtual-machine:~/Desktop/Assignment asweq
No such command: asweq
wyt11910104@wyt11910104-virtual-machine:~/Desktop/Assignment
```

- `touch` 创建文件:

```
wyt11910104@wyt11910104-virtual-machine: ~/Desktop/Assi...
wyt11910104@wyt11910104-virtual-machine:~/Desktop/Assignment/A3$ ./a.out
wyt11910104@wyt11910104-virtual-machine:~/Desktop/Assignment/A3 ls
a.out  cmake-build-debug  CMakeLists.txt  shell.c
wyt11910104@wyt11910104-virtual-machine:~/Desktop/Assignment/A3 touch test.out
wyt11910104@wyt11910104-virtual-machine:~/Desktop/Assignment/A3 ls
a.out  cmake-build-debug  CMakeLists.txt  shell.c  test.out
wyt11910104@wyt11910104-virtual-machine:~/Desktop/Assignment/A3
```

- `mkdir` 创建文件夹:

```
wyt11910104@wyt11910104-virtual-machine:~/Desktop/Assignment/A3 ls
a.out  cmake-build-debug  CMakeLists.txt  shell.c  test.out
wyt11910104@wyt11910104-virtual-machine:~/Desktop/Assignment/A3 mkdir test
mkdir: test: Directory is created successfully.
wyt11910104@wyt11910104-virtual-machine:~/Desktop/Assignment/A3 ls
a.out  cmake-build-debug  CMakeLists.txt  shell.c  test  test.out
wyt11910104@wyt11910104-virtual-machine:~/Desktop/Assignment/A3
```

- `rm` 删除文件/文件夹：

```
wyt11910104@wyt11910104-virtual-machine:~/Desktop/Assignment/A3 ls
a.out  cmake-build-debug  CMakeLists.txt  shell.c  test  test.out
wyt11910104@wyt11910104-virtual-machine:~/Desktop/Assignment/A3 rm test
rm successfully.
wyt11910104@wyt11910104-virtual-machine:~/Desktop/Assignment/A3 rm test.out
rm successfully.
wyt11910104@wyt11910104-virtual-machine:~/Desktop/Assignment/A3 ls
a.out  cmake-build-debug  CMakeLists.txt  shell.c
wyt11910104@wyt11910104-virtual-machine:~/Desktop/Assignment/A3
```