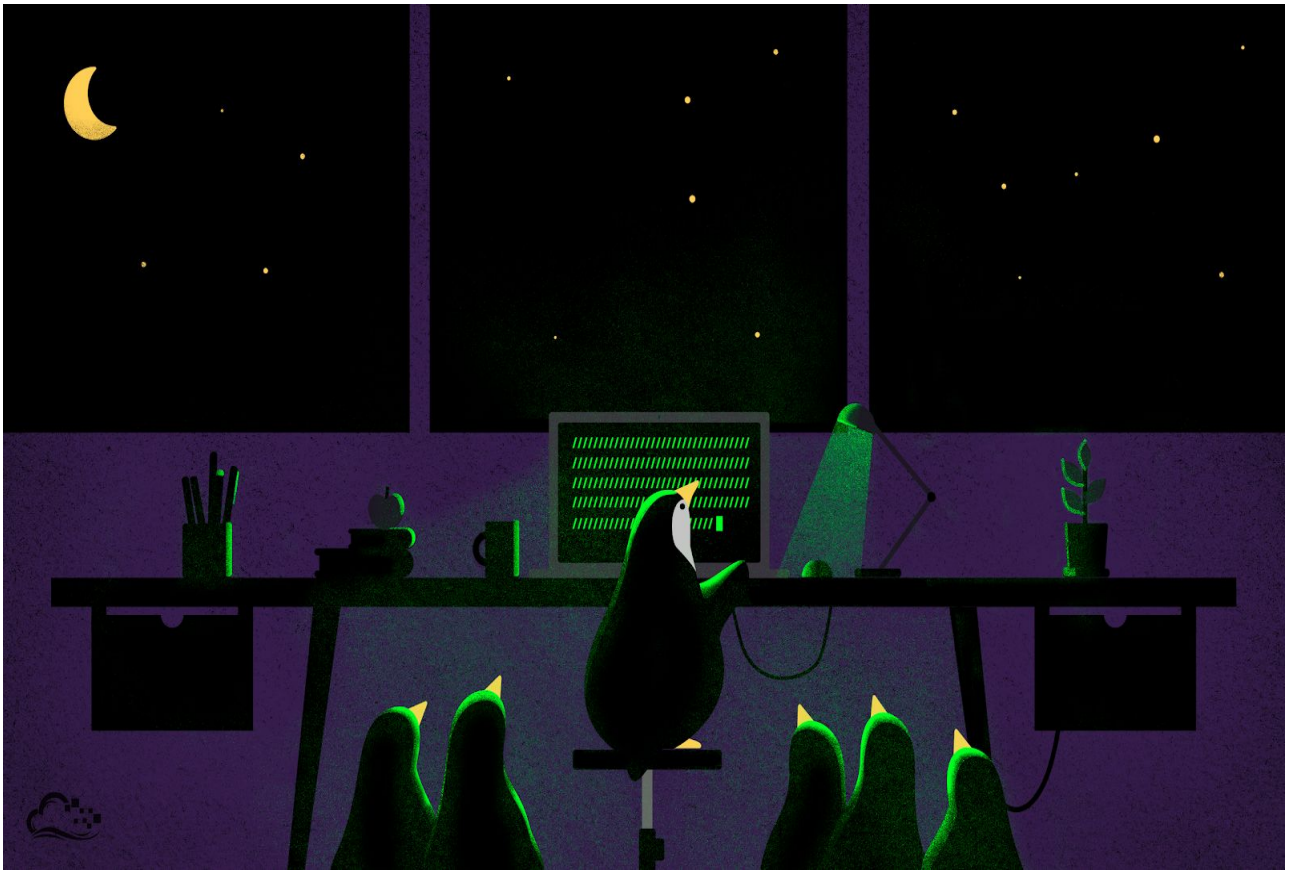
BÁO CÁO ĐỒ ÁN SIMPLE SHELL - HỆ ĐIỀU HÀNH



Danh sách thành viên

1. Võ Quốc Thắng 1712162

2. Lê Nguyễn Nhựt Trường 1712195

3. Lê Tuấn Đạt 1712329

09/09/2019

BỘ MÔN HỆ ĐIỀU HÀNH

Mục lục

1. Yêu cầu

- ❏ Thiết kế chương trình C đóng vai trò như giao diện Shell hỗ trợ đọc lệnh từ người dùng và thực hiện lệnh trong các tiến trình khác nhau.
- ❏ Nội dung của đồ án yêu cầu các phần: Tạo và thực hiện lệnh trong các tiến trình con, Hỗ trợ tiện ích xem lại lịch sử, chuyển hướng input và output, Hỗ trợ giao tiếp giữa tiến trình cha và con.

2. Thông tin

2.1. Thông tin thành viên

Họ và tên	MSSV	Email
Võ Quốc Thắng	1712162	voquocthangit@gmail.com
Lê Nguyễn Nhựt Trường	1712195	truongthk62014@gmail.com
Lê Tuấn Đạt	1712329	letuandat2110@gmail.com

2.2. Phân công công việc

STT	Công việc	Phụ trách
1	Tạo tiến trình con và xử lí lệnh trong tiến trình	Lê Tuấn Đạt
2	Cài đặt lịch sử	Lê Tuấn Đạt
3	Chuyển hướng input và output	Lê Nguyễn Nhựt Trường
4	Giao tiếp qua pipe	Võ Quốc Thắng
5	Test	Lê Nguyễn Nhựt Trường
6	Readme	Lê Tuấn Đạt

7	Viết báo cáo	Võ Quốc Thắng

3. Thiết kế chung

Lệnh nhận vào từ người dùng sẽ được phân tích thành các tham số đưa vào mảng arg sử dụng các hàm count_word, word_token và split.Chương trình nhận lệnh đến khi gặp phím "exit" Chương trình trước hết kiểm tra có xuất hiện lệnh "&" ở cuối dòng để thực hiện chạy đồng thời tiến trình cha với con. Nếu không, Chương trình cho tiến trình cha đợi đến khi tất cả các tiến trình con thực hiện chung.Kiểm tra các kí hiệu trong lệnh mà chương trình thực hiện hàm tương ứng.

4. Xử lý

4.1 Tạo tiến trình con và xử lý lệnh ở tiến trình con

Dùng hàm fork() để tạo tiến trình con quản lý bởi id. Tuy nhiên, hàm fork() mặc định sẽ cho tiến trình cha với con chạy đồng thời, vì vậy đối với trường hợp không có "&" ở cuối, phải để hàm wait(NULL) ở cha để tránh thực hiện đồng thời tiến trình cha và con.

```
//kiểm tra cuối lệnh có dấu "&", Nếu có đánh dấu concurrent = 0, nếu không
concurrent = 1
if (strcmp(args[len], "&") == 0)
{
        concurrent = 0;
        args[len] = NULL;
}
else concurrent = 1;

// Dùng hàm fork() để sinh tiểu trình con
pid_t id = fork();
if(id == 0)
{
   //Thực hiện các lệnh trong tiểu trình con
   exit(1);
 }
else if(concurrent == 1)  wait(NULL);
```

4.2. Tiện ích xem lại lịch sử

Sử dụng một biến previousCmd để lưu lại các lệnh trước (ngoại trứ các lệnh xem lịch sử) để khi có yêu cầu xem lại sẽ tiến hành thực hiện lệnh đó và trả lại kết quả cho người dùng. Biến history nhằm đánh giấu lệnh đang thực hiện là lệnh xem lại lịch sử để không lưu vào previousCmd.

```
if (strcmp(args[0], "!!") == 0)
{
        history = 0;
        if (previousCmd[0] == 0)
        {
        // Nếu không có lệnh trước thì in ra thông điệp
          printf("No command in history\n");
         continue;
         }
        else
        {
                                free(args);
                                strcpy(cmd,previousCmd);
                                printf("%s\n",cmd);
                                args = split(previousCmd, " ");
        }
}
else history = 1;
```

4.3 Chuyển hướng input và ouptut

Kiểm tra trong lệnh các ký hiệu "<" hoặc ">" để thực hiện lệnh tương ứng. Hàm dup2() hỗ trợ việc sao chép file description.

```
void filein(char **args)
{
        int i=0;
      while(args[i] != NULL)
```

```
void fileout(char **args)
{
          int i=0;
       while(args[i] != NULL)
```

```
    {
        if(strcmp(args[i],">") == 0)
        {
            int file_desc =
open(args[i+1],O_WRONLY|O_CREA
T);
            dup2(file_desc,1);
            while(args[i] != NULL)
            {
                args[i] = NULL;
                i++;
            }
            break;
        }
        i++;
    }
}
```

```
    {

        if(strcmp(args[i],"<") == 0)
        {

            strcpy(args[i],args[i+1]);
            args[i+1]=NULL;
            break;
        }

        i++;
    }
}
```

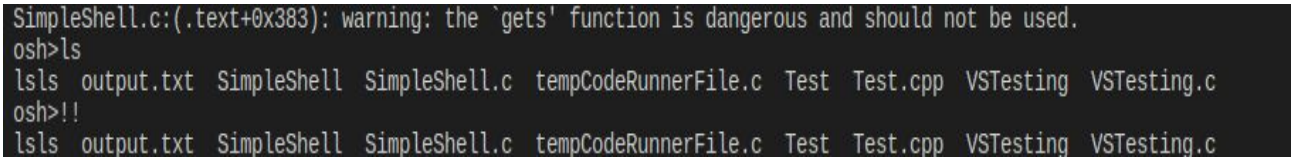
4.4 Giao tiếp giữa cha và con qua pipe

Sử dụng pipe để thực hiện giao tiếp giữa hai tiểu trình. Trong tiểu trình con tạo thêm một tiểu trình con khác thực hiện phần lệnh còn lại và tạo một pipe để tạo liên kết giữa hai tiểu trình này. Input của tiểu trình này là output của tiểu trình kia.

```
void communication(char **args)
{
        int i=0;
        int len = 0;
        while(args[len+1] != NULL) len++;
      while(args[i] != NULL)
      {

        if(strcmp(args[i],"|") == 0)
        {

          int pipefd[2];
          int childpid;
```

```
            pipe(pipefd);
            if(childpid=fork()){
              //parent
              //Write
              while(args[i]!=NULL)
              {
                args[i]=NULL;
                i++;
              }
                    filein(args);
                    fileout(args);
              close(pipefd[0]);
              dup2(pipefd[1],STDOUT_FILENO);
              execvp(args[0],args);
              wait(NULL);
              break;
            }
            else{
              //child
              //Read
              char ** subCmd = substr(args,i+1,len-i);
              close(pipefd[1]);
              dup2(pipefd[0],STDIN_FILENO);
              execvp(subCmd[0],subCmd);
              free(subCmd);
              exit(1);
            }
          }

          i++;
      }
}
```

5. Kết quả



6. Tổng kết

Qua đồ án này, chúng em đã hiểu thêm về cách vận hành của một terminal đơn giản. Biết thêm các lệnh sử dụng của UNIX cũng như ôn lại kiến thức đã học về ngôn ngữ C. Ngoài ra, đồ án còn mang lại cái nhìn khác hơn về môn học cũng như cách làm việc nhóm

REFERENCES

[1]. https://www.geeksforgeeks.org/
[2]. https://en.wikipedia.org/wiki/Unix
[3]. https://www.guru99.com/linux-pipe-grep.html