پیاده سازی SHELL

دانشگاه شیراز
زمستان 1399
سیستم های عامل
دکتر موسوی

زهرا محمد پور
9732497

مراحل پیاده سازی

دریافت ورودی و انجام pasre

ورودی در main از کاربر گرفته می‌شود و به تابع parse پاس داده می‌شود. خروجی تابع یک دوتای از اسم file وarg هایی است که باید به تابع execvp() داده شوند. در این تابع ورودی بر اساس فاصله و با جداسازی حالت های استثنا پردازش شده است.

```
def parse(inp):
```

پردازش دستور

اولین کار این است که دستور را با حالت های خاصی که با execvp() پیاده نشده مقایسه کنیم ودر صورت نیاز فراخوانی تابع هایی مثل kill را انجام دهیم.

```
        command = input("shimshell>>")
        if len(command) == 0:
            continue
        file, args = parse(command)
        flag = 0
        if command == 'exit':
            exit(0)
        elif command == "bglist":
            if len(bgList) < 1:
                print("no bg job")
            for i in range(len(bgList)):
                print("( " + str(i + 1) + " )" + convertStr(bgList[i]) )

        elif args[0] == "bgkill":
            if int(args[1]) - 1 < 0:
                print("invalid process number ")
            else:
```

در صورت ورود دستور cd یا gwd وارد تابع های مربوط شده و مقادیر ئن ها را بر میگردانیم.

```
def cd(path):
    return os.chdir(path)

def pwd():
    return os.getcwd()
```

حال نیاز است که عملیات fork را انجام دهیم زیرا میخواهیم پردازش جدید فرزند پردازش اصلی ما باشد و با اتمام آن shell بسته نشود. مقدار بازگشتی عملیات را در صورتی که پردازش bg باشد در یک لیست از p_id ها ذخیره میکنیم تا در عملیات های kill استفاده کنیم.

اگر مقدار بازگشتی عملیات fork برابر صفر باشد یعنی در پردازش فرزند هستیم و باید تابع execvp(file,args) فراخوانی شود.

اگر مقدار مثبت باشد در پردازش پدر هستیم و wait را صدا میزنیم تا زمانی که پردازش فرزند تمام شود صبر میکنیم و در صورت bg بودن فرزند به اجرا ادامه میدهیم.

مدیریت ارور هایی مثل ناموفق بودن fork یا execvp اینجا انجام میشود.

```
    p_id = os.fork()
    if flag == 1:
        bgId.append(int(p_id))

    if p_id == 0:
        a = os.execvp(file, args)
        if a < 0:
            print("eoror", file)
            os.exit(1)
    elif p_id < 0:
        print("Fork failed")
        exit(1)
    else:
        if flag == 0:
            status = 0
            os.waitpid(p_id, status)
        else:
            os.waitpid(p_id, os.WNOHANG)
```

تغییر رنگ

فایل config.txt شامل دو field رنگ برای background و foreground نوشته های درون کنسول است و توسط تابع getColor خوانده می‌شود. تابع changeColor رنگ را متناسب با کد تنظیم می کند.

```
def getColor():
    # Define a filename.
    filename = "config.txt"

    with open(filename) as f:
        content = f.readlines()
    bgline = content[0]
    fgline = content[1]

    bg = bgline.split(":")[1]
    fg = fgline.split(":")[1]

    return int(bg), int(fg)
def changColor():
    num1, num2 = getColor()

    print("\x1b[" + str(num1) + ";" + str(num2) + "m")
```

نگهداری دستور های قبل

با استفاده از کتابخانه ی readline دستوراتی که قبل تر در shell وارد شده ذخیره می شوند و با فشار دادن arrowKey up دیده میشوند.

زمان

با استفاده از کتابخانه ی date time ساعت هر بار و تاریخ بار اول نشان داده میشوند.