

# Hw2

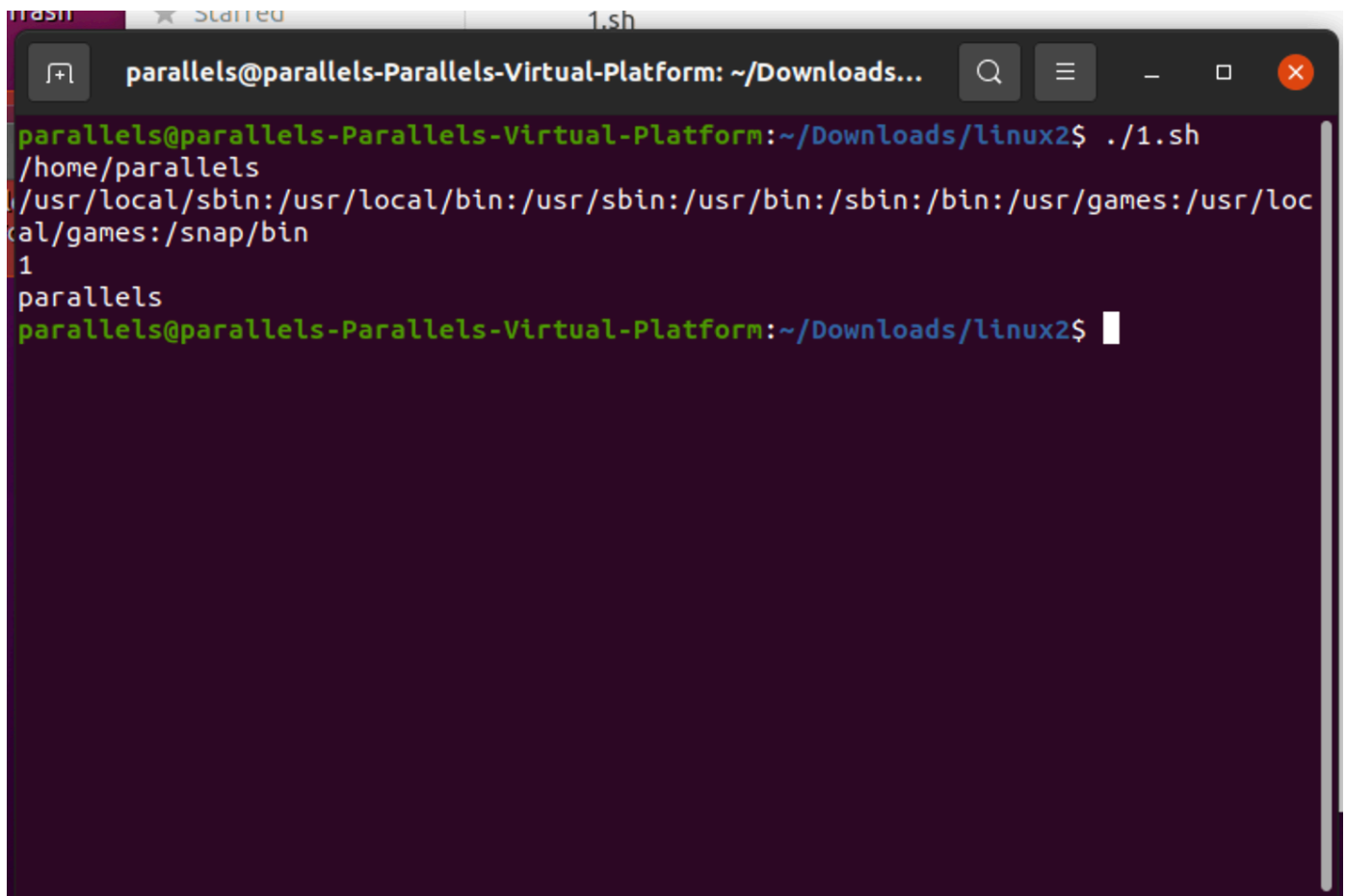
191250133 陶泽华

## 第一题

在一个sh文件中写入以下指令用来打印 HOME、PATH、SHLVL、LOGNAME 变量的值。

```
#!/bin/sh
# 输出HOME变量
echo $HOME
# 输出PATH变量
echo $PATH
# 输出SHLVL变量
echo $SHLVL
# 输出LOGNAME变量
echo $LOGNAME
```

运行结果如下（运行前需要赋给相应的权限 `chmod +x ./1.sh`）：

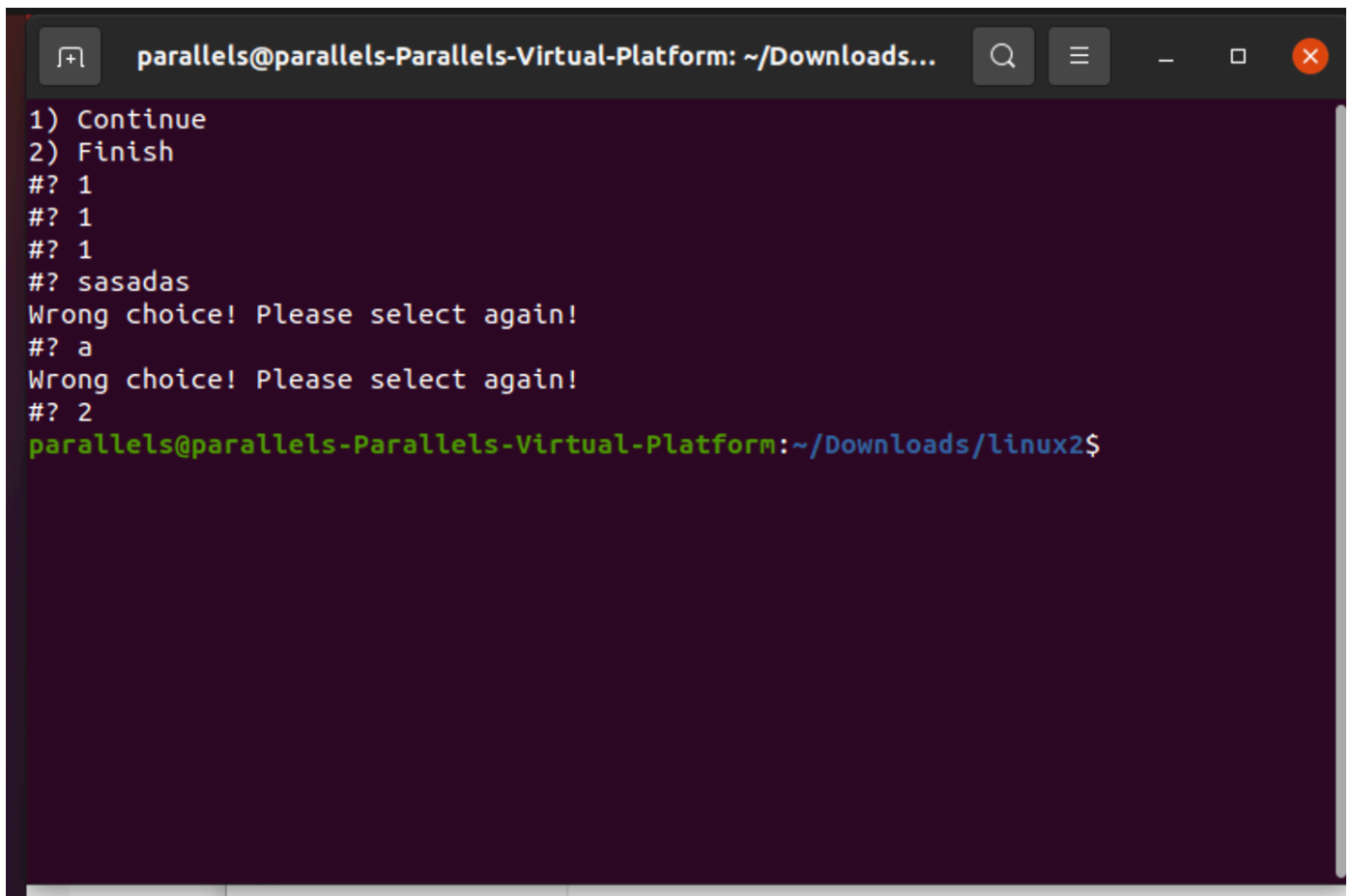
A terminal window titled "parallels@parallels-Parallels-Virtual-Platform: ~/Downloads..." shows the execution of a script. The prompt is "parallels@parallels-Parallels-Virtual-Platform:~/Downloads/linux2\$". The command ". /1.sh" is entered. The output is: "/home/parallels", "/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin", "1", and "parallels". The prompt then returns to "parallels@parallels-Parallels-Virtual-Platform:~/Downloads/linux2\$".

```
parallels@parallels-Parallels-Virtual-Platform:~/Downloads/linux2$ ./1.sh
/home/parallels
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
1
parallels
parallels@parallels-Parallels-Virtual-Platform:~/Downloads/linux2$
```

## 第二题

```
#!/bin/sh
clear
select item in Continue Finish
do
case "$item" in
    Continue) ;;
    Finish) break ;;
    *) echo "Wrong choice! Please select again!" ;;
esac done
```

运行结果如下：



```
parallels@parallels-Parallels-Virtual-Platform: ~/Downloads...
1) Continue
2) Finish
#? 1
#? 1
#? 1
#? sasadas
Wrong choice! Please select again!
#? a
Wrong choice! Please select again!
#? 2
parallels@parallels-Parallels-Virtual-Platform: ~/Downloads/linux2$
```

1. `#!/bin/sh` 是一个约定的标记，它告诉系统这个脚本需要/bin/sh解释器来执行，即使用哪一种shell。
2. `clear` 用来进行清屏
3. `select in` 是 Shell 独有的一种循环，格式如下：

```
select variable in value_list
do
    statements
done
```

这里的 `select item in Continue Finish` 中的 Continue、Finish 这两个变量可以看做一个取值列表。当运行到 `select` 时，取值列表中的内容会以菜单的形式显示出来，用户输入菜单编号，就表示选中了某个值，这个值就会赋给变量 `item`，然后再执行循环体中的 `statements`。

4. `case...esac` 中的语句表示若用户选择1就代表continue，继续执行下去；若用户选择2就代表finish，会执行break语句跳出循环；若用户输入其它，代表无一匹配模式，使用星号\*捕获该值，再执行后面的命令，就会输出Wrong choice! Please select again!

## 第三题

```
=====Makefile1=====
# 设置变量，方便后续使用
# shell pwd用来获取相对路径，必须在pwd前面加shell，然后把shell pwd当一个变量来引用
export Top:=${shell pwd}
export Src:=$(Top)/src/
export Include:=$(Top)/include/
export Build:=$(Top)/build/

all:
# @可以避免将当前正在执行的命令输出/回显到屏幕上
# 这个指令用来执行指定位置$(Src)的makefile
@$(MAKE) -C $(Src)

install:
# 复制$(Build)/test文件到变量$(Top)对应的目录下
@cp $(Build)/test $(Top)

clean:
# 删除$(Build)和$(Top)/test目录下的文件及目录
@rm -rf $(Build) $(Top)/test

===== Makefile2=====
# main.o test4.o:执行这个target需要的参数prerequisites，也就是生成target所需要的先决条件,比如所需要的文件或是目标;
all:main.o test4.o
# 执行的步骤
# 递归创建$(Build)目录，即使上级目录不存在，会按目录层级自动创建目录
@mkdir -p $(Build)
# 将prerequisites移动到$(Build)目录下
@mv *.o $(Build)
# 执行$(Src)/dir1、$(Src)/dir2目录下的makefile
$(MAKE) -C $(Src)/dir1
$(MAKE) -C $(Src)/dir2
# (CC)是一个全局变量，在终端使用gcc作为编译器，将$(Build)/test后的所有文件进行静态链接，生成$(Build)/test可执行文件
$(CC) -o $(Build)/test $(Build)/*.o $(Build)/dir1/*.o $(Build)/dir2/*.o

main.o : $(Include)/func.h
# 将main.c进行编译但是不链接，并且指定头文件路径$(Include)
$(CC) -c main.c -I$(Include)
```