Лабораторная работа
Написание сценариев.

1 Специальные параметры

Специальные параметры трактуются оболочкой особым образом. Имена специальных параметров состоят из одного символа. Значения этих параметров можно только читать.

`*` Раскрывается в строку позиционных параметров начиная с первого. Если `*` используется внутри двойных кавычек, то преобразуется в одно слово, состоящее из позиционных параметров разделенных первым символом переменной `IFS`. Если `IFS` – NULL, то параметры объединяются, если `IFS` не установлена, то параметры разделяются пробелами.

Раскрывается в строку позиционных параметров начиная с первого. Если используется внутри двойных кавычек, то каждый позиционный параметр преобразуется в отдельное слово.

`#` Число позиционных параметров в десятичном виде.

`?` Статус завершения последнего выполненого конвейера.

`-` Список опций заданных в командной строке, установленных встроенной командой set или самой оболочкой.

`$` ID процесса оболочки. В подоболочках `$` сохраняет значение и выдает ID процесса оболочки, а не подоболочки.

`!` ID процесса последней выполненной в фоновом режиме команды.

`0` Имя оболочки или сценария.

2 Условные операторы

```
if список1; then список2;
[ elif список3; then список4; ]
...
[ else список5; ]
fi
```

Выполняется *список1* если его код завершения ноль, то выполняется *список2*. Иначе выполняются поочередно все конструкции `elif`. Если *список3* завершается с кодом ноль, то выполняется *список4* и команда завершается. В противном случае выполняется *список5*. Код завершения равен коду завершения последней команды или нулю если ни одно из условий не было выполнено.

```
case слово in
[(] шаблон [ | шаблон ] ... ) список ;;
...
esac
```

Команда сначала раскрывает слово и ставит его в соответствие каждому шаблону. Когда совпадение найдено выполняется соответствующий список. После первого совпадения попытки найти соответствующий шаблон прекращаются. Код завершения равен нулю если найти совпадение не удалось и коду завершения последней выполненой команды в противном случае. Для проверки соответствия слова шаблону используются те же правила, что и для имен файлов.

Для выполнения команд по условию можно, также, использовать списки:

```
[ -f $1 ] || { echo 'File not exist!'; exit 1;}
[ ! -f $2 ] && cp $1 $2
[[ -x /bin/sh || -x /bin/bash ]] \
        && echo Found || echo Not found
```

В первой строке проверяется существование файла указанного первым аргументом сценария. Если файла не существует, то выводится сообщение и сценарий завершается. Во второй строке проверяется отсутствие файла указанного вторым аргументом. В последней строке проверяется существование хотя бы одного из файлов sh и bash и выводится соответствующее сообщение.

3 Комментарии

В неинтерактивном режиме оболочка воспринимает слова начинающиеся с символа # и все остальные символы до конца строки как комментарии и игнорирует их. В интерактивном режиме комментарии, как правило, не допускаются.

4 Практическое задание

1. Проанализируйте файл `/home/labs/back.sh`
2. В подкаталоге `bin` создайте файл `case.sh`:

```
case $1 in
start) if [ -f /tmp/back$UID.pid ] ; then echo Alredy started
       else /home/labs/back.sh >/tmp/back$UID.pid & \
          echo Started: `cat /tmp/back$UID.pid`
       fi ;;
stop)  if [ -f /tmp/back$UID.pid ] ; then \
          kill -kill `cat /tmp/back$UID.pid` && echo Killed
          rm /tmp/back$UID.pid
       else
       echo Not started
       fi ;;
status) [ -f /tmp/back$UID.pid ] \
           && echo Running: `cat /tmp/back$UID.pid` \
           || echo Not running ;;
*) echo "usage: case.sh {start|stop|status}"
esac
```

3. Выполните команду `export UID`
4. Запустите сценарий без аргументов и последовательно с аргументами status, start, status, start, stop, status, stop. Результат сохраните в файле отчета.
5. В подкаталоге `bin` создайте файл `calc.sh`:

```
res=$1
shift
while [[ $# -ge 2 ]] ; do
   res=$(($res$1$2))
   shift 2
done
echo $res
```

6. Запустите сценарий следующими командами:

```
a) calc.sh 1 + 2 \* 3 + 3 / 4]
b) calc.sh 1 + 2 \* 3
c) calc.sh 1 + 2\*3
d) calc.sh 1+2
```

7. Объясните почему различается результат в случаях b) и c).

8. Исправьте сценарий так, чтобы в случае d) он выдавал результат 3.

9. Напишите сценарий `summ.sh` выводящий сумму своих аргументов если первый аргумент +, произведение если первый аргумент и сообщение `Usage: summ.sh +|* arguments ...` в противном случае. Например:

```
bash-2.04$ summ + 4 8 3
15
bash-2.04$ summ \* 4 2 3
24
```