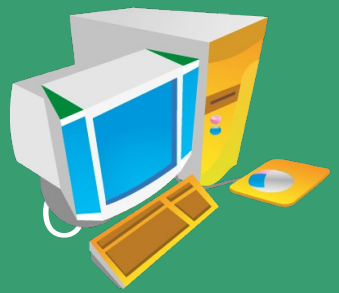
Гуляев Г.М.

Операционная система Linux и СПО

Занятие 5. Взаимодействие программ

Курс по переобучению на использование СПО



Текстовые потоки

- **Утилиты в Unix поддерживают три стандартных текстовых потока: стандартный вход (stdin), стандартный выход (stdout) и поток ошибок (stderr)**
- **Соглашения по умолчанию: stdin идет с клавиатуры терминала, stdout и stderr идут на монитор терминала**
- **Программа, которая умеет работать с stdin, stdout и stderr, называется фильтром: stdin --> программа --> stdout,stderr**
- **Благодаря соглашению о поддержке текстовых потоков, программы Unix умеют взаимодействовать друг с другом, передавая данные одна другой**
- **Здесь каждую новую программу не нужно писать полностью с нуля (как в Windiows), так как можно использовать уже существующие программы**
- **Стандартные потоки имеют номера: 0 - stdin, 1 - stdout и 2 - stderr. В командах можно явно выбирать поток, указав его номер**

Перенаправление ввода/вывода

- **Символ > задает перенаправление вывода в файл:**

  $ ls > flist

  $ echo Привет! > flist

- **Символ >> дописывает поток вывода в конец файла:**

  $ pwd >> flist

- **В качестве файла может быть устройство (принтер, модем и др.). Специальное устройство /dev/null служит для погашения текстового потока**

  $ cat flist > /dev/null

- **Утилита wc считает количество слов (wc -l - строк) в тексте**

- **Символы < и << задают перенаправление ввода:**

  $ wc -l < flist (вывод на экран количества строк в файле flist)

- **Можно сочетать перенаправления:**

  $ wc -l < f1 > f2 (запись в файл f2 количества строк в файле f1)

  $ wc -l > f2 < f1 (то же самое)

Конвеер команд

- **Средство, объединяющее стандартный выход одной команды со стандартным входом другой, называется конвеером и задается вертикальной чертой |**
  - **$ ls | wc -l (число строк в списке файлов текущего каталога)**
  - **$ cat flist | wc -l (число строк в файле flist)**
- **Утилита less умеет организовать постраничный показ текста**
  - **$ ls /etc | less (листание клавишами PgUp/PgDn, q — выход)**
  - **$ history | less**
- **Утилита grep умеет искать в тексте нужные строки**
  - **$ ls | grep f**
  - **$ dmesg | grep usb**
- **Утилита sort сортирует строки по возрастанию (sort -r по убыванию)**
  - **$ sort flist или sort < flist (сортировка строк в файле flist)**
- **Конвейером можно объединять много команд:**

$ cat f1 | grep группа | sort | cat -b > f2 (в файле f2 отсортированные и пронумерованные строки файла f1, содержащие слово группа)

Перенаправление ошибок

- **Утилита tr меняет одни символы на другие**

  date +%D | tr "/" "."

  echo $PATH tr ":" "\n"

  echo 3 8 7 5 1 4 6 2 | tr " " "\n" | sort | tr "\n" " "; echo

- **Утилита cut вырезает нужные столбцы**

  cut -d<разделитель> -f<столбцы>

  столбцы задаются перечислением 2,5,7 или 3-8

  echo 3 8 7 5 1 4 6 2 | tr " " "\n" | sort | tr "\n" " "; echo

- **$ cat f1 f2 (вывод f1 и f2 на экран, при отсутствии f2 будет ошибка)**

  $ cat f1 f2 2>/dev/null (подавление вывода ошибок)

  $ cat f1 f2 2>err.txt (перенаправление потока ошибок в файл)

- **Более сложные случаи перенаправления:**

  $ cat f1 f2 >>ff 2>ff (stderr и stdout в один файл ff)

  $ cat f1 f2 2>>ff 1>&2 (stderr в ff и stdout в stderr, то есть в ff)

Группировка команд

- **Условное выполнение команд:**

  && выполнение последующей команды при условии нормального завершения предыдущей, иначе игнорировать

  || выполнение последующей команды при ненормальном завершении предыдущей, иначе игнорировать

- **Последовательное выполнение команд в строке:**

  $ date; pwd; ls

- **Для группировки команд также могут использоваться фигурные {} и круглые () скобки. Примеры:**

  k1 && k2; k3 (k2 выполняется если выполнилась k1, k3 - всегда)

  k1 && {k2; k3} (k2 и k3 выполняются если выполнилась k1)

- **Круглые скобки (), кроме выполнения функции группировки, выполняют еще функцию вызова нового экземпляра интерпретатора shell:**

  (cd ..; ls) ls (список файлов родительского каталога, затем текущего)

Запуск сценариев

- **Создадим текстовый файл cfile с командами:**

  date

  pwd

  ls

- **Запустить на выполнение такой сценарий можно, передав его аргументом интерпретатору:**

  $ sh cfile

  $ bash cfile

  $ sh < cfile

- **Удобнее сделать сценарий исполняемым, тогда он не будет отличаться от любой другой программы Linux:**

  $ chmod 711 cfile (установка прав rwx x x)

  $ ./cfile (запуск сценария)

- **Команды в сценарии могли быть записаны в одной строке:**

  date; pwd; ls

Запуск сценариев

- **Таким образом, выполняемыми файлами могут быть не только файлы, полученные в результате компиляции и сборки, но и файлы, написанные на языке shell (командные сценарии)**
- **Во избежания проблем с выбором интерпретатора в сценариях принято в первой строке указывать путь на исполняемый файл интерпретатора:**

  #!/bin/bash

  Знак # указывает, что данная строка является комментарием
- **Язык интерпретатора bash достаточно развит (за основу был взят язык Algol) и имеет все необходимые конструкции (операторы, условия, циклы, массивы и др) для написания программ любой сложности**
- **Благодаря умению Unix - программ взаимодействовать друг с другом, shell-программирование быстро и успешно решает многие задачи, для которых в системах типа windows потребовалось бы создание трудоемкого и сложного софта**

Пример: вычисление выражений

- **Утилита bc умеет разбирать и вычислять выражения со скобками и математические функции (bc -l). Вычисления могут проводиться с любой точностью**
- **Примеры:**

  $ echo "3/4" | bc -l

  $ echo "25*30" | bc -l
- **Создадим сценарий calc.sh:**

```
#!/bin/bash
echo "scale=50;"$1 | bc -l
```

$1 - параметр к сценарию: $ calc.sh <выражение>

- **Примеры:**

  $ calc.sh "3/4"

  $ calc.sh "25*30"

  $ calc.sh "(162.162/2.25+0.828)/0.0125/(5.1*5.1+10.2*3.9+3.9*3.9)"

Выполнить самостоятельно

1. Составить команду, которая создавала бы в текущем каталоге подкаталог с именем "Мой каталог" и, в случае успеха, делала бы его текущим и копировала бы в него все файлы из родительского каталога

2. Выполнить по очереди две команды

echo 1111 > test.txt; cat test.txt > /dev/null || echo 2222

echo 1111 > test.txt; cat test.txt > /dev/null && echo 2222

Объяснить результаты (почему так получилось?)

3. Составить команду для вывода на экране всех строк некоторого текстового файла подряд сначала все по возрастанию, затем все по убыванию (текстовый файл предварительно создать)

4. Составить команду, которая при выполнении операции копирования файлов сообщения об ошибках добавляла бы в конец файла errors.txt и при этом создавала бы этот файл, если его еще нет

5. Составить команду для ввода текста в файл с клавиатуры, который сразу же после создания перемещался бы в подкаталог moved при наличие в тексте слова move

6. Составить команду для вывода бы на экран числа пользователей, имеющих в файле /etc/passwd ссылку на шелл вида: /bin/sh

Вопросы для самоконтроля

1. Какие текстовые потоки обрабатывают утилиты Unix-систем?
2. Каким образом выходящий текстовый поток направить в файл?
3. Как в качестве входящего текстового потока взять содержимое файла?
4. Как поток ошибок направить в файл?
5. Каким образом можно дописать выходящий поток в конец файла?
6. Как передать выходящий текстовый поток другой программе?
7. Что такое конвеер команд?
8. Как погасить выходящий текстовый поток?
9. Как погасить поток ошибок?
10. Назовите операторы условного выполнения команд?
11. Как проверить существует ли файл?
12. Какую первую строку должен содержать командный сценарий?

Спасибо за внимание !

www.altailand.ru