Операционные системы (6 семестр)

Лекция 2.2. Отложенное и регулярное выполнение заданий.
Лекция 2.3. Текстовые файлы и потоки (часть 1)

Ассистент, к.т.н. Митричев Иван Игоревич

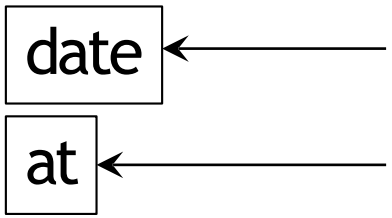
Москва 2018

План лекции

- Отложенное выполнение заданий: команды at, atq.
- Регулярное выполнение заданий: служба cron.
- Перенаправление потоков ввода/вывода. Конвейеры и фильтры.
- Просмотр файлов с помощью more и less.
- Команды head и tail.
- Вырезание текста с помощью cut.
- Сравнение файлов и каталогов.

Отложенное выполнение заданий

Цели отложенного выполнения – установить однократное автоматическое выполнение задачи.



- вывести дату и время;
- выполнить команду в определенный момент времени.

Указание времени: now + <n> minutes, hours, days, weeks, month, day, HH:MM, midnight, noon, teatime (4pm), pm, am, today, tomorrow.

Команда at ставит задания в очередь (обслуживается службой atd):
at now +5 minutes
> ls
> ps
[Ctrl-D]

Настройка отложенного выполнения заданий

`at <время> -f <filename>` ← • указать команды в файле;

`atq` или `at -l` ← • получить список заданий;

`atrm <номер задания>` ← • удалить задание из списка;

atq - обычный пользователь получает список из собственных заданий, суперпользователь получает список заданий всех пользователей.

`/etc/at.allow`
`/etc/at.deny`
← • списки пользователей, кому можно и кому нельзя пользоваться командой at;

`batch <команда>` ← • выполнить команду, когда средняя загрузка системы уменьшится до значения 0.8 (/proc/loadavg).

Регулярное выполнение задач

Регулярное выполнение предназначено для регулярных задач: обновление системы, проверка дисков, сканирование компьютера на наличие вредоносных программ и т.д.

Для регулярного выполнения задач используется служба cron.

В различных версиях GNU/Linux используются разные пакеты cron.

•/etc/crontab – общесистемная таблица заданий;

•/var/spool/cron – каталог с пользовательскими таблицами периодических заданий.

Файл системной таблицы заданий /etc/crontab

```
SHELL=/bin/sh
PATH=/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
# runparts
02 4 * * * root runparts /etc/cron.daily
22 4 * * 0 root runparts /etc/cron.weekly
42 4 1 * * root runparts /etc/cron.monthly
```

Поля 1-5 – указание периода выполнения задания. Шестое поле – пользователь, с правами которого выполняется задание. Седьмое поле – исполняемая команда.

Файл crontab

Последняя строка файл системной таблицы заданий /etc/crontab

```
42 4 1 * * root runparts /etc/cron.monthly
```

[время] [пользователь] [команда]

Задание будет запускаться в 4 часа 42 мин. первого числа каждого месяца.

Формат задания времени:

- минуты часа (0-59);
- час суток (0-23);
- число (1-31);
- месяц (1-12);
- дни недели, начиная с воскресенья (0-6).

crontab ← • операции с таблицами заданий.

Таблицы crontab пользователей

```
echo '30 21 * * * echo hello' > crontab1
crontab crontab1
```

← • установить таблицу заданий;

Создан файл crontab1, содержащий строку с заданием. Таблица crontab1 считана и установлена взамен старой таблицы.

`crontab -l` ← •таблица заданий текущего пользователя;

`crontab -l -u <username>` ← • таблицу заданий указанного пользователя;

Редактирование таблицы заданий

`crontab -e` ← • редактирование текущей таблицы заданий в текстовом редакторе.

Таблица откроется в текстовом редакторе по умолчанию.

`crontab -r -u <username>` ← • удалить таблицу заданий для пользователя;

```
$ crontab -r
$ crontab -l
crontab: no crontab for user1
```

Перенаправление потоков ввода/вывода

При открытии файла создается его дескриптор - неотрицательное целое число. С этим числом связывается поток ввода/вывода в файл/из файла. Для любого процесса ядро Unix/Linux всегда создает три стандартных потока:

- поток ввода (stdin), файловый дескриптор 0;
- поток вывода (stdout), файловый дескриптор 1;
- поток вывода ошибок (stderr), файловый дескриптор 2.

Поток ввода всегда открыт только на чтение, когда как поток вывода и поток ошибок – только на запись. Поток ввода по умолчанию связан с клавиатурой, потоки вывода и ошибок – с дисплеем.

Можно использовать следующие операторы перенаправления потоков:

< – оператор перенаправления стандартного потока ввода;

> или 1> – операторы перенаправления стандартного потока вывода;

2> – оператор перенаправления стандартных потоков ошибок.

Версия операторов перенаправления без перезаписи файла: >>, 1>>, 2>>

Пример: `ls 2>&1` ← перенаправление потока ошибок в поток вывода, & - совмещение потоков

Перенаправление потоков в файл

`set -o noclobber`	• запрет перезаписи файлов при перенаправлении;
`set +o noclobber`	• сброс опции noclobber;
`set -o`	• просмотр опций.

>| – перенаправление потока вывода с гарантированной перезаписью файла;
2>| – перенаправление потока ошибок с гарантированной перезаписью файла;
>> и 2>> – перенаправление потоков вывода и ошибок с добавлением к файлу.

```
$ set -o noclobber
$ ls -l > ls.txt
bash: ls.txt cannot overwrite existing file
$ ls -l >| ls.txt
```

В данном примере файл 1.txt будет перезаписан, несмотря на то, что установлена опция оболочки noclobber.

Конвейеры и фильтры

Конвейер (pipe) позволяет объединить поток вывода одного процесса с потоком ввода другого процесса.

`ps aux | view -` ←

- организация конвейера (| – символ конвейера);

список входивших в сеанс пользователей передан команде просмотра view, нак "дефис" сообщает, что текст должен быть получен из потока ввода, в который передает свой вывод команда last;

`ps | tee ps.txt` ←

- выводит информацию из потока ввода в файл(ы) и в поток вывода (tee – пример фильтра).

Команда tee используется для отладки сложных конвейеров.

Фильтр –это команда, обрабатывающая данные. и пропускающая их далее по конвейеру.

more, less – постраничный просмотр текстовых файлов. Общее название - пейджеры (pager). По умолчанию для просмотра страниц man используется именно less, что можно изменить при помощи опции man -p <pager> или переменной окружения MANPAGER.

Команды less и more позволяют просматривать несколько файлов.

Команды head, tail, wc

`head -<n> <filename>` ← • вывести n первых строк файла или потока (по умолчанию 10);

`tail -<n> <filename>` ← • вывести n последних строк файла или потока;

`tail -f <filename>` ← • динамически выводит последние строки файла, удобно следить за журналами.

Пример: `ps -a | tail`

```
wc /etc/passwd
38   60 1948 /etc/passwd
```
← • подсчет строк, слов, символов;

`wc -l <filename>` ← • подсчет строк в текстовом файле.

```
$ wc text*
 2   3  20 text
 3   3  18 text1
 4   3  16 text.1
 1   3  18 text2
20  15  80 text.2
 4   6  36 text3
20  15  80 text.3
 3  15 132 text.5
57  63 400 итого
```

Вывод текста с помощью cut

cut <опции>

- выводит только заданные символы, байты или поля из файла.

Требуемые для вывода символы строки указывают после опции -c через запятую или тире.

Опции: b – байты;
c – символы; f – поля;
d – символ – разделитель полей.

Примеры:

```
$ ls -ld
drwx------ 51 user1 user1 3752  Oct 22 21:04 .
$ ls -ld | cut -c1-10,35-43
drwx------ 3752
```

Разделитель полей команды cut по умолчанию – TAB.

cut -f1,3 d: /etc/passwd

- вывести список пользователей системы и их идентификаторы UID.

Использование списков в опциях команды cut

Список состоит из байтов, символов или полей.
2 – второй байт, символ или поле;
2-5 – все байты, символы и поля со второго по пятый;
-3 – все до четвертого;
5- – начиная с пятого;
1,3,6 – первый, третий и шестой байты, символы или поля.
1,3- – первый и все байты, символы или поля, начиная с третьего.

Примеры:

```
$ cat cut.txt
1       2       3
4       5       6
$ cut -f2- cut.txt
2       3
5       6
```

```
$ cat cut.txt
cut command
w   command
awk command
wc  command
$ cut -d " " -f2 cut.txt
command

command
```

- символ пробел, как разделитель

http://rus-linux.net/MyLDP/consol/cut.htm l

Сравнение файлов и каталогов

```
diff <опции> <объект1> <объект2>
```

- сравнить файлы или каталоги.

```
diff ps1.txt ps2.txt > servicepack.patch
patch ps1.txt servicepack.patch
```

- обновление файлов;

Файлы-"заплатки" patches создаются с помощью утилиты diff, а обновление файлов производится утилитой patch.

```
cmp <file1> <file2>
```

- побайтное сравнение файлов;

```
diff3 <file1> <file2> <file3>
```

- сравнение трех файлов.

Проверка идентичности (целостности) файлов может быть произведена командой sum или командами вычисления хэш-функций md5sum, sha256sum и другими.

Контекстное сравнение файлов

diff -c file1.txt file2.txt

Результат

***** file1.txt 2018-02-21 17:10:29.764650235 +0300**
--- file2.txt 2018-02-21 17:10:50.768329841 +0300
****
***** 1,4 ******
яблоки
- ананасы
груши
морковь
--- 1,4 ----
яблоки
груши
морковь
+ петрушка

file1.txt:
яблоки
ананасы
груши
морковь

file2.txt:
яблоки
груши
морковь
петрушка

Специальные символы в выводе команды

! - указывает, что эта строка первого файла заменена на другую во втором файле;

+ - указывает строку во втором файле, которую необходимо добавить в первый файл.

- - указывает строку в первом файле, который необходимо удалить, чтобы получит второй файл.

Унифицированный режим сравнения файлов

file1.txt:
яблоки
ананасы
груши
морковь

file2.txt:
яблоки
груши
морковь
петрушка

Унифицированный режим (опция -u) аналогичен контекстному, но он не отображает лишнюю информацию.

diff -u file1.txt file2.txt

Результат:

--- file1.txt 2018-02-21 17:10:29.764650235 +0300
+++ file2.txt 2018-02-21 17:10:50.768329841 +0300
@@ -1,4 +1,4 @@
яблоки
- ананасы
киви
морковь
+ петрушка