

UNIX/Linux системы в инфокоммуникациях

Продвинутый пользовательский курс в UNIX/Linux системах

В. Титов, О. Садов
`tit@linux-ink.ru`

Университет Информационных технологий, Механики и Оптики
кафедра Телекоммуникационных Систем

23.11.2015

Удалить все процессы, выполняющие заданную команду.
Вывести PID удаляемых процессов или сообщение об
отсутствии таких процессов. Перед собственно удалением
процессов запрашивать подтверждение на эту операцию.
Параметр имя команды.

Скрипт 1:

```
#!/bin/bash
usage () {
    echo ; echo "Скрипт должен иметь один параметр."; echo
    echo ; echo "      Вызов:"; echo
    echo "      $0 <имя команды>"; echo
    exit
}
if test $# -ne 1; then usage; fi

LIST2REMOVE='ps -C $1| grep -v 'PID TTY'| awk '{ print $1 }'
echo $LIST2REMOVE
for process in $LIST2REMOVE; do
    ps a | grep -e " *$process"
    echo "Kill proccess $process?"
    read ch
    if test $ch != "y"; then
        kill $process; fi
done
```

Написать скрипт, создающий структуру домашних директорий в директории, определенной вторым параметром (если он есть) или в текущей директории. Вторым параметр задает файл с именами тех пользователей, для которых создается такая структура в формате `passwd`. Собственником созданных директорий должны быть соответствующие пользователи. То же относится к группам. Проверить запущен ли скрипт пользователем `root`.

Скрипт 2:

```
#!/bin/bash
usage () {
    echo ; echo "$1"; echo
    echo "        Вызов:"; echo
    echo "        $0 <имя файла с пользователями> <директория>"
    exit }

if test $# -lt 1; then
usage "Скрипт должен иметь по крайней мере один параметр"
fi

FILE=$1; shift; DIR=.

if ! test -f $FILE; then usage "Нет такого файла: $FILE"; fi
pushd `dirname $FILE`>/dev/null; FULLNAME=`pwd`/`basename $FILE`

if [ $# -ge 1 ]; then
    DIR=$1
    if ! test -d $DIR; then usage "Нет такой директории: $DIR"
    pushd $DIR
fi
```

Скрипт 2:

```
USER='id|cut -d\( -f1|cut -d= -f2'
if ! test $USER == 0; then usage "Скрипт запущен не пользователем root"
for usr in `cat $FULLNAME`; do
    NAME='echo $usr|cut -d: -f1'
    USR='echo $usr|cut -d: -f3'
    GR='echo $usr|cut -d: -f4'
    HDIR=$(basename $(echo $usr|cut -d: -f6))
    echo "mkdir $HDIR; chown $USR:$GR $HDIR"
done
if [ "$DIR" != "." ]; then
    popd
fi
```

Организовать простейший обмен сообщениями между двумя компьютерами. используя протокол SSH.

Для организации обмена сообщениями вам может понадобиться получить доступ к другому компьютеру по ssh.

```
$ ssh 172.16.41.104
```

Первый раз ответить yes.

Можно организовать доступ по ssh без пароля.

```
$ ssh 172.16.41.104; yes,
```

```
$ ssh-keygen -t dsa (ввести парольную фразу)
```

В ~/.ssh появятся два файла: id_dsa, id_dsa.pub

```
$ cat id_dsa.pub >> authorized_keys2
```

```
$ chmod go-rw ~/.ssh/authorized_keys2
```

```
$ ssh 172.16.41.104, ввести парольную фразу,
```

```
$ ps ax | grep ssh-agent
```

```
$ ssh-add id_dsa
```

```
$ ssh 172.16.41.104
```


Что нужно сделать, чтобы установить новое программное обеспечение?

- Настроить (возможно, применив патчи)
- Скомпилировать
- Установить (разместить файлы в нужных местах)
- `configure`
- `make`
- `make install`

- Легкость использования (rpm вместо configure/make/make install)
- Ориентирование на понятие «пакет»
- Возможность обновления пакетов (см. след. пункт)
- Отслеживание межпакетных зависимостей (в отличие от MSW, где ОС управляет приложениями с точки зрения всей системы)
- Возможность различных запросов
- Возможность верификации
- Поддержка различных процессорных архитектур
- Использование «чистых» исходников

RPM-пакет

RPM пакеты поставляются в виде сжатых архивов, которые содержат не менее 1 файла, а также инструкции по установке этих файлов, включая права доступа, которые должны быть применены к каждому файлу в процессе установки. Эти инструкции также могут содержать скрипты, которые запускаются перед или после установки (удаления) пакета. Имя пакета делится на **четыре** части, отделяемые друг от друга дефисами или (последние две части) точками:

имя-версия-релиз.архитектура.rpm. например:

`kernel-2.6.32-279.5.1.el6.i686`

`kernel-headers-2.6.32-279.14.1.el6.i686`

Имеется два типа пакетов: бинарные пакеты и пакеты с исходным кодом.

`kernel-2.6.32-279.5.1.el6.src`

Программа/система управления пакетами (rpm) ведет базу данных всех установленных в системе пакетов. Она находится в директории `/var/lib/rpm`.

Управление пакетами

| | | |
|---|-----|-------------|
| Обновление/установка | -U | --upgrade |
| Установка | -i | --install |
| Удаление | -e | --erase |
| Режим запросов | -q | --query |
| Верификация | -V | --verify |
| Проверка подписи | -K | --cheking |
| Обновление в режиме freshen | -F | --freshen |
| Инициализация БД | Нет | --initdb |
| Перестройка БД | Нет | --rebuilddb |
| Установите/удалите пакет (/var/lib/tftp...) | | |

```
# rpm -U foo-0.1-1.i386.rpm
```

```
# rpm -e foo
```

Установлен ли пакет, номер версии:

```
rpm -q foo
```

Вместе с опцией -q можно также использовать другие опции запроса.

```
rpm -qa foo
```

```
rpm -qa | grep foo
```

Информация о пакете:

```
rpm -qi foo
```

Список файлов пакета

```
rpm -ql foo
```

Список пакетов, от которых зависит этот пакет:

```
rpm -qR foo
```

Можно управлять форматом вывода информации:

```
rpm -q kernel \  
    --queryformat="%{NAME} %{RELEASE} %{SOURCERPM}\n"
```

```
rpm -qa --last | head
```

```
rpm -V initscripts
```

Запросы: --requires, --provides, --whatrequires,
--whatprovides, , --scripts, --triggers.

```
yum [options] [command] [package ...]
```

Команды yum:

- `install` — используется для установки последней версии пакета или группы пакетов, обеспечивая удовлетворение всех взаимных зависимостей пакетов;
- `update` — при запуске без указания имени пакета обновляет все установленные в системе пакеты, если указано имя пакета, обновляется только указанный пакет;
- `remove` или `erase` — используется для удаления из системы указанных пакетов;
- `list` — используется для просмотра информации об имеющихся пакетах;
- `info` — получение информации о пакете.
- `provides|whatprovides` — зависимости.
- `groupinstall|groupinstall|groupremove|groupinfo`.

- `localinstall` — используется для установки пакета с локального компьютера. При этом будут автоматически найдены и установлены пакеты, необходимые для разрешения зависимостей;
- `localupdate` — используется для обновления пакета с локального компьютера;
- `deplist` — показывает список имеющихся репозиториев. По умолчанию показываются только разрешенные репозитории.
- `replist` — показывает список имеющихся репозиториев. По умолчанию показываются только разрешенные репозитории.
- `search` — поиск по всем доступным репозиториям.

Yum, опции

| | |
|------------------------|--|
| -h, --help | Справка |
| -t, --tolerant | Режим игнорирования не критичных ошибок |
| -C | Работа из кеша (не обновлять его) |
| -c config_file | Альтернативный конфиг-файл |
| -R minutes | Максимальное время ожидания в минутах |
| -d debug_level | Уровень отладочных сообщений |
| -e error_level | Уровень многословности сообщений |
| -y | Автоматически отвечать "yes" |
| --version | Вывести версию yum |
| --installroot=path | Указать корневой каталог установки, отличный от каталога по умолчанию (корневой) |
| --enablerepo=repo | Сделать доступным доп. репозитории |
| --disablerepo=repo | Сделать недоступными репозитории |
| -x, exclude=package | Исключить пакет по имени или шаблону |
| --noplugins | Отключить плагины yum |
| --nogpgcheck | Отключить проверку подписи gpg |
| --disableplugin=plugin | Отключить конкретный плагин по его имени |

- yumex,
- PackageKit,
- Nautilus.

- Планирование того, что желательно собрать (в качестве образца можно использовать уже имеющийся `src.rpm`)
- Комплектование ПО для пакета (архивы с исходным кодом, патчи)
- Накладывание при необходимости патчей на исходный код
- Осуществление воспроизводимой сборки ПО (`configure/make/make install`)
- Планирование обновлений (как обновлять/удалять версии пакета)
- Отслеживание зависимостей (определить зависимости)
- Сборка пакета
- Тестирование пакета

Дерево директорий:
<rpmbuild>/

{SOURCES,SPECS,BUILD,RPMS/{noarch,i686},SRPMS,BUILDROOT}
Поместить исходный код (.tgz) в SOURCES, спец-файл —
в SPECS.

Секция общей информации:

Summary:

Summary (ru):

%define version 1.1

Licence:

Group: Development/Languages

Name: xxx

Prefix: /usr

Provides: xxx

Release: 1

Source: xxx-%version.tar.gz

URL: <http://ibm.com/developerworks/opensource/jikes>

Version: %version

Buildroot: /tmp/xxx

%description

%description -l ru

Секция prep:

```
%prep  
%setup -q  
%patch
```

Секция build:

```
%build  
./configure --prefix=$RPM_BUILD_ROOT/usr ...  
make
```

Секция install:

```
%install  
make install
```

Секция clean:

```
%clean  
rm -rf $RPM_BUILD_ROOT
```

Секция files:

```
%files
%defattr(-,root,root)
/usr/bin/xxx
%doc /usr/doc/xxx-%{version}/license.htm
%doc /usr/man/man1/xxx.1*
```

Секции post, preun, postun:

Команда rpmbuild

`rpmbuild -bСтадия_ сборки имя_пакета.spec`

- ba Собрать бинарный пакет и пакет с исходным кодом
- bb Собрать бинарный пакет
- bc Скомпилировать программу, но не собирать rpm-пакет, то есть выполнить до секции %build включительно
- br Выполнить подготовку и остановиться сразу после завершения стадии %prep
- bi Выполнить сборку бинарного пакета и остановиться сразу после завершения стадии %install
- bl Выполнить проверку списка файлов для пакета и вывести резюме ошибок, если корневой каталог сборки не содержит каких-то файлов из списка
- bs Собрать только пакет с исходным кодом


```
rpmlintl <файл пакета.rpm>
```

Скрипты

```
rpm -qa | grep rpm
```

```
#!/bin/sh
```

```
tid='rpm -q --qf "%{INSTALLTID}\n" $*'
```

```
rpm -q --tid $tidrpm -qa | grep rpm
```