Дата последней сборки документа:
19 мая 2020 г. в 03:01

Практика использования и наиболее полезные конструкции командной оболочки bash

Содержание

1. Сценарии командной оболочки bash

Для того чтобы успешно создать и запустить сценарий командной оболочки требуется:

1. Написать сценарий,
2. Сделать сценарий исполняемым,
3. Поместить сценарий в каталог, где командная оболочка сможет найти его.

Простой пример в качестве иллюстрации

bash_test.sh

```
#!/bin/bash
echo "This is number of command line args: $#"
```

Теперь нужно сделать этот файл исполняемым

```
$ chmod +x bash_test.sh
$ ls -l bash_test.sh
# выведет
# -rwxr-xr-x 1 ADM 197121 31 май 19 02:55 bash_test.sh*
```

Теперь можно вызывать этот скрипт

```
$ ./bash_test.sh 10 20
# выведет
# This is number of command line args: 2
```

2. Конструкции оболочки bash

Найти в корневом каталоге и всех подкатлогах (`/`), обычные файлы (`-type f`), измененные за последний день (`-mtime -1`), за исключением тех файлов, у которых есть суффикс `.o` (`! -name '*.o'`)

```
find / -type f -mtime -1 ! -name '*.o'
```

Вывод имен файлов и удаление файлов с именами `core` или `junk` из рабочего каталога и всех его подкаталогов (круглые скобки обязательно отделяются пробелами)

```
find . \( -name core -o -name junk \) -print -exec rm {} \;
```

Скопировать все csv-файлы из родительской директории (..) в текущую (.)

```
cp -ip ../*.csv
```

Скопировать файл из родительской директории в текущую директорию

```
cp -ip ../Cheat_sheet_Git/cheat_sheet_git.tex .
```

Скопировать одну директорию в другую

```
cp -rip ../Cheat_sheet_Git/style_packages/ .
```

Переименовать файл

```
mv cheat_sheet_git.tex cheat_sheet_bash.tex
```

Найти все файлы с расширением *.csv и выбрать из них те, в которых содержится строка 'state' (для каждого файла, отвечающего поисковому шаблону, запускается свой процесс)

```
find . -name '*.csv' -exec grep -niE 'state' {} \;
```

Вывести список файлов из текущей директории и всех поддиректорий

```
ls -l *
```

Найти среди файлов с расширением *.py те, в именах которых есть подстрока 'spark' (используется конвейер)

```
ls -l *.py | grep -iE 'spark'
```

Найти файлы с расширением *.py и к каждому из них применить команду grep, которая будет искать в файле подстроку 'argparse' без учета регистра, с выводом номера строки, на которой она нашла искомую строку по регулярному выражению 'argparse' (работает медленно, так как для каждого файла, отвечающего поисковому шаблону запускается свой процесс)

```
find . -maxdepth 1 -name '*.py' -exec grep -inE 'argparse' {} \;
```

Альтернативный вариант с использованием xargs (работает значительно быстрее варианта с -exec)

```
find . -maxdepth 1 -name '*.py' | xargs grep -inE 'argparse'
```

Найти в файлах с расширением *.tex строку 'section' без учета регистра и вывести три строки контекста

```
find . -name '*.tex' | xargs grep -iE 'section' -3
```

Вывести список пакетов, в именах которых встречается подстрока 'python' с контекстом 'sql'

```
conda list | grep -inE 'python.*sql'
```

Получить информацию о доступном метсе на диске

```
df -h
```

Скачать файл с тем же именем, что на удаленном репозитории

```
curl -O http://merionet.ru/yourfile.tar.gz
```

Скачать файл с удаленного репозитория с новым именем и/или путем

```
curl -o newfile.tar.gz http:// merionet.ru /yourfile.tar.gz
```

Возобновить прерванную загрузку с того места, где она остановилась

```
curl -C - -O http://merionet.ru/yourfile.tar.gz
```

Скачать несколько файлов

```
curl -O http://merionet.ru/info.html -O http://wiki.merionet.ru/about.html
```

2.1. Переадресация ввода-вывода

Перенаправить *стандартный поток вывода* данных (дескриптор файла 1) и *стандартный поток вывода ошибок* (дескриптор файла 2), которые возвращает `conda` с захватом всех пакетов, в именах которых встречается подстрока `'python'`, в файл с именем `test_file.log` (временный поток вывода данных `&1`). Если команда вернет ошибку, то сообщение ошибки перепишет содержимое файла `test_file.log`

```
conda list | grep -inE 'python' > test_file.log 2>&1
```

Более короткий вариант рассмотренной выше конструкции

```
conda list | grep -inE 'python' &> test_file.log
```

Присоединить стандартный поток вывода данных и стандартный поток вывода ошибок к содержимому файла. Конструкция `rm df` возвращает сообщение об ошибке «`rm: cannot remove 'df': No such file or directory`», которое можно добавить в файл

```
rm df &>> test_file.txt
```

Вывести размер директорий в Мегабайтах (`M`)

```
du -BM
```

Вывести итоговый размер директории (`c`) в Мегабайтах (`M`)

```
du -cBM
```

Перевести размеры директорий в понятный человеку формат

```
du -h
```

Вывести информацию об указанной директории в дружественном формате

```
du -sh style_packages/
```

Вывести размер папок текущей директории, не погружаясь глубже корневых папок

```
du -h -d 1
```

или так

```
du --max-depth=1 -h
```

Вывести информацию об использовании дискового пространства иерархией каталога с подсчетом общего размера каталога и перенаправлением стандартного потока вывода ошибок в «черную дыру»

```
du -ch -d 1 2>/dev/null
```

Список литературы

1. Собель М. Linux. Администрирование и системное программирование. 2-е изд. – СПб.: Питер, 2011. – 880 с.