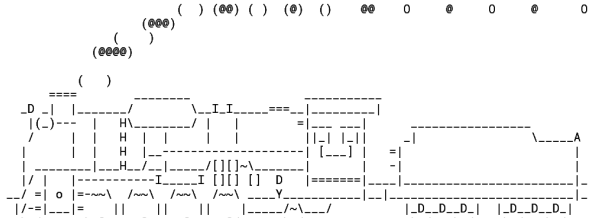
Операционни системи

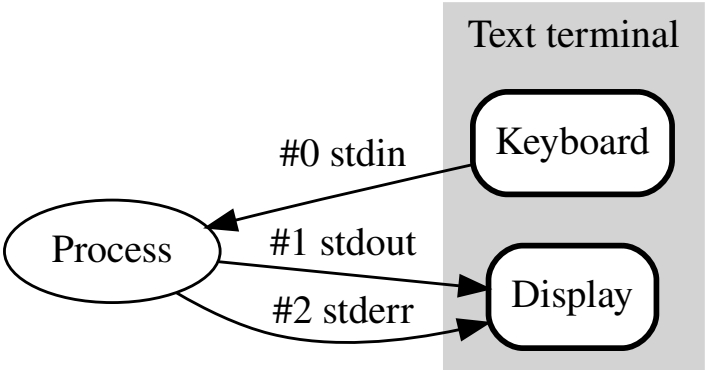
ФМИ

СИ 2018

Обработка на текстови файлове



Стандартни стриймове



- stdin - стандартен вход
- stdout - стандартен изход
- stderr - стандартен изход за грешки

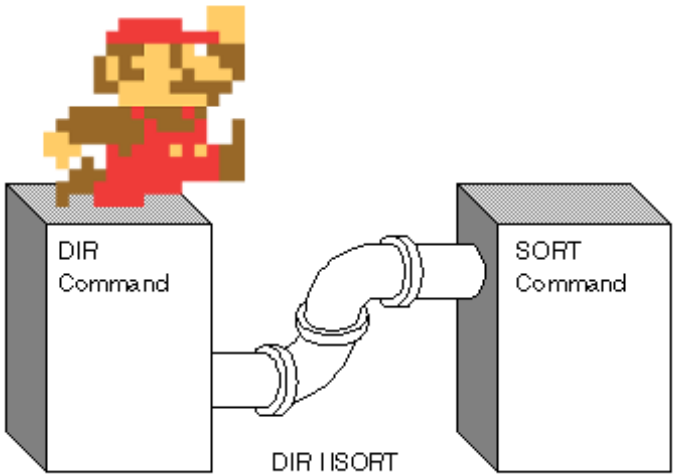
Пренасочване на I/O

Стриймовете могат да се пренасочват

- Пренасочване на вход:
  - < - прави се с този символ
  - `$ command < file`
  - `$ sort < /etc/password`
- Пренасочване на изход:
  - > - презаписва файла с резултата на командата
  - `$ command > file`
  - `$ echo "something" > file`
  - >> - добавя в края на файла
  - `$ command >> file`
  - `$ echo "something else" >> file`
  - The 1 is silent - 1>

- Пренасочване на изход за грешки
  - 2> - 2 вече не се подразбира
  - `$ command 2> log_file`
  - `$ ls -alR /proc/ 2> /dev/null`
- Пренасочване на stdout и stderr към един и същи файл
  - `$ command > file 2>&1` - пренасочва се изходния поток, а после насочваме този за грешки към него
  - `$ ls -R /proc/ > output 2>&1`
  - `$ command &> file`
  - `$ ls -R /proc/ &> output`

Pipe



Pipe

- Пайповете позволяват стандартния изход на една програма (тази вляво) да се подаде като стандартен вход на друга (тази вдясно)
  - `$ ls -al /proc | less`
  - `$ cut -d: -f6 /etc/passwd | sort | uniq -c | sort`
- Могат да се съчетават с пренасочване - обикновено се пренасочва изходът за грешки към стандартния
  - `$ ls -R /proc 2>&1 | grep 'kernel'`

Команди за работа с текст

- `$ cat foo bar baz` - конкатенира подадените ѝ файлове
- `$ tac foo bar baz` - прави същото като cat, но ги обръща
- `$ paste foo bar` - показва файловете един до друг, разделени с tab (или с друг разделител, подаден като параметър)
- `$ paste foo bar baz | tac`

Команди за работа с текст

- `$ wc foo` - изпринтва броя на новите редовете, думите и байтовете във файла
- При подаване на няколко параметъра изкарва статистика за всеки от тях и обща статистика
- `-c; --bytes`
- `-m; --chars`
- `-l; --lines`
- `-w; --words`

Команди за работа с текст

- `cut` - извлича и принтира избраните части от подадения му файл
- Ако не са подадени аргументи, чете от стандартния вход
- Използва tab като разделител по подразбиране, но може да се подаде друг (`-d:`)
- `$ cut -c 2-4 foo` показва само от втория до четвъртия символ от всеки ред на файла foo
- `$ cut -d @ -f 1 foo` - например, може да покаже само името в един имейл
- Адски е удобен, ако имаме структуриран текст, например колонки от текст

Команди за работа с текст

- `$ tr` - заменя, свива и/или трие символи от стандарния вход, пишейки на стандарния изход
- заменяне
  - `$ tr a-z A-Z` - ще замени всички малки английски букви с големи
- свиване
  - `$ tr -s '\n'` - ще замени всяка поредица от подадения символ само с един
  - `$ echo "baaaabaaaaa" | tr -s a`
- триене
  - `$ tr -d '\000'` - трие всички срещания на подадения аргумент
  - Използва се често за премахване на специални символи
  - `echo bakbak | tr -d k`

Търсене в съдържание на файл

- `$ grep` - търси шаблони вътре в съдъжанието на файлове
- `$ grep -n` - показва на кой ред се е срещнал шаблонът
- `$ grep -F` - търси за низ вместо за регулярен израз
- `$ grep -i` - case insensitive
- `$ grep -v` - показва тези редове, които **не** изпълняват шаблона
- `$ grep -n bash /etc/passwd`
- Може да се подават регулярни изрази

Stream **ED**itor

- `$ sed` - сед приема входен текст, прави указаните трансформации с него и изптринтва резултат
- Удобен е за някои манипулации, за които не си заслужава да включвате текстови редактор
- Може да приема вход от pipe, file или stdin
- `$ cat family.txt | sed 's/Baba/Dqdo/g'`

AWK

- Тюринг complete език за програмиране
- Специално предназначен за обработка на текст и извличане на информация
- Може да разделя редовете като cut
- Може да използва регулярни изрази като grep
- Поддържа математически операции
- `$ awk -F ':' '$1~"baba" {print $2}' family.txt`

Други полезни команди

- `$ sort` - не е изненадващо какво прави
- По подразбиране сортира по първата колона, но това може да се смени. Както и разделителя между поленцата
- По подразбиране сортира лексикографски
  - `1 2 232 3`
- `$ sort -n` - може да сортира и по големина на числата
  - `1 2 3 232`
- `$ sort -r` - може да сортира наобратно
- Може да се подават няколко файла, които да сортира (и смеси)
- Често се ползва, за да се приготви текста за командата **uniq**

Други полезни команди

- `$ uniq` - премахва дублиращи се последователни редове в сортиран текст
- `$ uniq -c foo` - показва всеки ред и брой срещания на реда
- Резултатът на `$ uniq -c` може да се подаде на `$ sort -r` и по този начин да се получи сортиране по най-често срещан ред

Текстови файлове++

- Често командите могат да взимат няколко файла като параметър
- Досадно е тези имена да се въвеждат ръчно
- Линуксаджиите не обичат да пишат излишно, затова:
  - Wildcard chararcter - това е единствен символ, например (*), който замества няколко символа или празен стринг
  - glob (programming) - шаблон, с който се специфицират няколко файла наведнъж чрез wildcard символи

Специални wildcard символи

- ? - намира срещане на един символ
- * - намира срещане на произволен брой символи
- `[...]` - намира срещане на класове от символи
  - `[bar]`
  - Може да приема интервали с `-`
  - `[a-z]`, `[a-z2-5]`

Употреба на `{}`

- Използват се за генериране на стрингове
- `{a,u}` се оценява до `a u`
- `b{a,u}ba` се оценява до `baba buba`
- `colo{,u}r` се оценява до `color colour`
- Може да се съчетава с wildcard символи
- `mv foo.{txt,md}`

Специални символи

- Метасимволи - \, ?, *, $, (,), {, }, [,], \
- Екраниране с \
- " " - екранират всичко, освен \, $, \
- ' ' - екранират всичко, освен \
- “Екраниращa последователност (на английски: Escape sequence), е поредица от символи, използвани за промяна на състоянието на компютрите.”
  - из българската Уикипедия

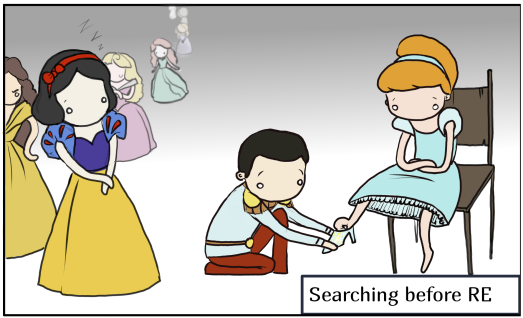
Влагане на команди

- `$ command1 $(command2)` - подава като аргумент на първата команда изхода от втората
- `$ command1 `command2``
- Двете нямат семантична разлика, но `$()` е по-мощно (може да се влага)

Изпълняване на няколко команди

- Могат да се изпълняват по няколко последователни команди
  - отделят се с ;
  - `$ touch foo; cat foo; echo "Not empty" > foo; cat foo`
- Дългите команди могат да се разделят на няколко реда
  - прави се с \
  - `$ ls \-l`
  - `$ touch foo; \> cat foo; \> echo "stuff" > foo; \`
    `> cat foo`

Регулярни изрази



Регулярни изрази

- Механизъм за извличане на шаблони
- Сложни (понякога)
- Важни
- Могат да се използват в grep, sed, perl,...
- `egrep`
- Според POSIX има модерни (egrep) и базови(ed).
- For more info `man up`
- `regex(7)`

Регулярни изрази

- Повечето букви и цифри се оценяват на себе си
- Има специални символи, които се интерпретират по специален начин
- Специалните символи могат да се екранират

Регулярни изрази и специални символи

- `\t` - таб
- `\n` - нов ред
- `\r` - carriage return
- `\f` - form feed
- `\x` - hex символ
- `.` - един произволен символ
- има още доста

Регулярни изрази++

Има специални символи, които указват край/начало на дума/ред

- ^RE - начало на ред
- RE$ - край на ред
- \<RE - начало на дума
- RE\> - край на дума

Регулярни изрази++11

Могат да се използват цели класове от символи, които се заграждат в `[...]`

- `[...]` - match-ва кой да е символ от изброените
- Има някои предефинирани класове - `foo[ab[:class:]cd]bar`
  - `[:upper:]` - `[A-Z]`
  - `[:lower:]` - `[a-z]`
  - `[:alpha:]` - `[a-Z]`
  - `[:digit:]` - `[0-9]`
  - `[:alnum:]`, `[:xdigit:]`, `[:punct:]`, `[:blank:]` (само TAB и space), `[:space:]`...
- Може да се използва `-` за рейндж. Например, `[a-c]`
- `[^..]` - ще match-не всеки символ, който не е сред изброените

Регулярни изрази - примери

- `$ grep '[[:upper:]]' /etc/passwd`
- `$ egrep '^[rb]' /etc/passwd`
- `$ egrep '^[^rb]' /etc/passwd`

Регулярни изрази++ 20

KEEP
CALM
WE'RE
ALMOST
THERE

Регулярни изрази++ 20

Трябва да можем да указваме и брой срещания

- `*` - 0 или повече срещания
- `+` - 1 или повече срещания
- `?` - 0 или 1 срещане
- `{n}` - точно **n** срещания
- `{n,}` - поне **n** срещания
- `{n,m}` - между **n** и **m** срещания
- Винаги ще се опитва да хване възможно най-много (greedy)

Регулярни изрази епизод 8

Резултатите, които се match-ват, могат да се групират и преизползват

- `(RE)`- прави група
- Към тази група може да се обърнем с `\n`, където n е номерът на групата (номерират се подред от 0). Т.е., ако имаме (RE1)(RE2)...(REn), то `\1` ни дава RE1... `\n` ни дава REn (n > 0)
  - `(.)\1` - (.) хваща произволен сивмол, запазва го в група едно и целият израз и match-ва два поредни еднакви символа
  - `(a)\1` ще хване всички срещания на ‘aa’
- `(R1|R2)` - R1 or R2
  - `$ egrep (baba|dqdo) foo`
- abc{3} vs. (abc){3}

Регулярни изрази - дебел пример

```
$ cat foo > Parenthesis allow you to store matched patterns.

$ sed -r s/(.)\1/\[\1\1\]/g foo
> Parenthesis a[ll]ow you to store matched pa[tt]erns.
```

Обработка на текст

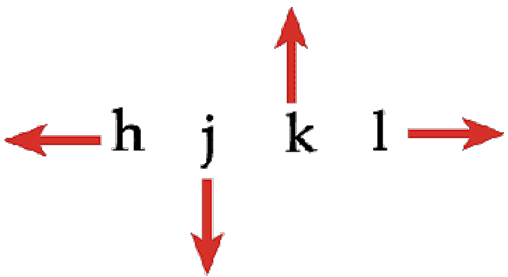
- Изключително важно за Unix философията
- “Дръж ми бирата и глей к’во мога” - всичко, което реално е нужно, е текстов редактор
- Много възможни редактори - изборът е религиозен въпрос
- `$EDITOR` - контролира редактора по подразбиране
- **Nano**, **vi**, **Emacs**

vi/vim

- vi - The **Vi**sual Editor
  - Достъпен под UN(*)X
  - Разработен като **ex** за BSD Unix от Bill Joy през 1976
  - Пуснат като **vi** през 1979
- vim - **vi** i**m**proved
  - Пуснат през 1991
  - Много повече функционалност
  - vi compatible mode
- neovim
  - Пренаписване от нулата на vim, по-красив и компактен код
  - Започнат през 2014
  - бързи плъгини (lua, python), по-лесно скриптване

Vi 101

- Modes
  - Normal (Command) mode - всичко се интерпретира като команда
  - Insert mode - За въвеждане на текст в документ (влиза с с `i` преди текущия символ или `a` - след текущия символ)
- Придвижване



- Основни комади в command mode
  - dd - трие ред
  - x - трие символ
  - yy - копира ред
  - p - paste
- Save & Exit
  - :w - save
  - :q - quit (if saved or no changes)
  - :q! - quit without save
  - :wq, :x - save and quit

- Движения
  - `w` - една дума
  - `$` - до края на реда
  - `^` - до началото на реда
  - `G` - до края на файла
  - `gg` - до началото на файла
  - `tx` - до символа `x`
- Комбиниране с командите
  - `yw` - копира една дума
  - `y3w` - копира три думи
  - `5dd` - трие 5 реда
  - `d$` - трие до края на реда
  - `dtf` - трие всичко до символа `f`
  - `cG` - изтрива текста до края на файла и влиза в insert mode

- vimtutor
- openvim.com
- vim-adventures.com