

# 计算机系统

## 3. 编程



# 评论

- 计算机、信号、信息、它们的表现
- 数字的表示方法
- 固定点，浮动点表示
- 编码写作
- ASCII, UTF-8代码表
- 架构，操作系统的作用
- 公务员-服务器的差异
- 图形和字符连接
- 文件系统，基本命令

\_\_\_\_\_

# 接下来的事情.....。

- 进一步的命令
- 前景和背景中的进程
- I/O重定向
- 过滤器
- 让我们开始编程吧!
- 正则表达式!

修改后：2020年。  
10.08.

## 计算机系统

---

页码：3

# 进一步说明？

- 命令在前台：正常的命令执行
- 命令在后台执行：命令末尾的&字符
  - 和命名：安培尔，和，和符号，阿特符号
- **Sleep 15 #** 等待15秒，在此期间，不可能运行新的命令。
  - CTRL+Z向进程发送一个停止信号
- **sleep 15 & #** 睡眠进程在后台运行，在此期间我们可以发出新的命令
  - 结果我们得到。[1] 28321，第一个数字是 "jobnumber"，第二个数字是PID（processid）。
- 工作--列出我们正在运行的命令（我们的 "工作"）。
- 通过ps-process status命令，我们也可以看到后台运行的睡眠。



# 在后台的进程

- 要把一个后台进程替换到前台，使用：`fg %1`
- 在停止一个前台进程后（用`ctrl-z`），我们也可以在后台继续它：`bg %1`
- 发出一个信号：`kill-signal`进程
  - 进程由它的工作编号（必须使用%符号）或其pid（进程标识符）编号来识别。



```
illes@valerie:~$  
illes@valerie:~$ sleep 25 &  
[1] 28345  
illes@valerie:~$ kill -SIGSTOP %1  
illes@valerie:~$ ps  
  PID TTY          TIME CMD  
27724 pts/4    00:00:00 bash  
28345 pts/4    00:00:00 sleep  
28351 pts/4    00:00:00 ps  
  
[1]+  Stopped                  sleep 25  
illes@valerie:~$ bg %1  
[1]+  sleep 25 &  
illes@valerie:~$  
[1]+  Done                      sleep 25  
illes@valerie:~$
```

# 更多关于流程的内容

- **top**命令：你可以看到关于运行中的进程的数据，全局系统的状态
- 每个人都是平等的？
- Unix、Linux系统优先级在-20、19之间（40级）。
- 数字越小越优先！
- **nice**命令，修改了要启动的命令的优先级。
- 聋哑人的优先级是0，顶部或**ps -l**命令NI column!
- **nice -n 5 sleep 20& #** 新的优先级将是5（它将5加到0）。
- **nice -n -10 sleep 20& #** 只有拥有root权限的人才能给予更多

的优先权!!!!!!。

修改后：2020年。  
10.08.

## 计算机系统

---

页码：6

# 流程、优先事项

- Unix和Linux根据优先级来处理进程!
- POSIX4-IEEE1003.1b (1993) , 出现了实时扩展。
- 两个优先列表
  - 漂亮的-20 - +19, 正如我们看到的那样
  - 实时优先级列表, 数值在0-99之间 (100个优先级) 。
    - 在这个列表中, 最大的数字意味着最大的优先权
  - 这两份名单的共同表述如下。
    - 99,98...1,-20,-19,...0,1,...19,通常它被称为140优先级区间, 其中可以有不同的映射方式

修改后：2020年。  
10.08.

## 计算机系统

---

页码：7

# 延迟执行命令

- 在Unix系统中，延迟执行有两种不同的支持方式
  - Cron(demon)，需要超级用户权限，或者在 `/etc/cron.allow` 中拥有一个权限。
    - `/etc/crontab -system timing, /etc/cron.d-user timings`
    - 这些时间决定哪个程序在哪个时间必须由cron启动
  - 如果atd服务正在运行，可以使用At - 命令。
    - 命令等待来自标准输入的输入（要运行的命令）。
    - 在现在+5分钟<命令文件#在需要的时候，我们可以给很多方式
    - 在0845年10月1日<also # also将在8.45日执行。

# 要完成一个命令

- 正常完成--命令不想继续运行
- 一个命令最大限度地运行到该时间，而它的启动用户是登录的！
  - 不管它是前台还是后台进程，都是一样的。
- **Nohup命令**
  - Nohup命令& # 它可以不使用&，但在这种情况下，它是一个前台进程！
    - 命令的输出将被保存在nohup.out文件中！
    - 这个文件是在启动的时候创建的，在注销之后，输出将被附加到这个文件中。
    - 人们可以定义一个自己的输出文件：nohup命令 > userdefined.nohup

修改后：2020年。  
10.08.

# 计算机系统

---

页码：9



# 撇号

- 在命令行中，合法的字符是。.,\_-,数字,普通字符
- 撇号：",","--修改字符的经典含义
- 在单引号中，'每个特殊的影响，例如：特殊的/\$%等。影响消失
- 例如：`echo 'apple$tree' #apple/$fa`
- 在双引号中，"\$、a/、a`和'字符的影响仍然存在。
  - 例如：`a=RealMadrid; echo "Let's go $a!"# 让我们去RealMadrid!`
- \x字符：修改x的原始含义
  - 例如：`tree=flower; echo apple\ $tree #apple$tree`
  - `Echo apple$tree #appleflower`



# 输入和输出的重新定向

- 输入、输出设备

- `stdin(0)` - 键盘，标准输入通道（默认输入）。
- `stdout(1)` - 监视器，标准输出通道（默认输出）。
- `stderr(2)` - 监控，标准错误通道（默认错误输出）。

- 重新定向

- 输出：在符号`>`的帮助下
  - 例如：`echo 苹果 桃子 李子> 水果`
  - `Echo apple >&2` #并非是为名为2的文件。

- 输入：在符号<的帮助下

- 例如：passwd steve <apple; cat apple # appletree, appletree

修改后：2020年。  
10.08.

计算机系统

---

页码：11

# I/O重定向 II.

- 输出，附加
  - >>文件名，例如：`echo nut >>fruit`
  - 如果它不存在，那么它就会被创造出来!
- 错误-输出(stderr)重定向
  - `2>, 2>>`
- 为了对称起见()`<<`
  - 输入重定向，同时没有得到<<后给出的文本（现在是苹果）。

修改后：2020年。  
10.08.

## 计算机系统

页码：12

```
$ cp 2>myerror  
$ mv this 2>>myerror  
$ cat myerror
```

```
猫<<苹果  
<input type=text name=X>  
<input  
type=button> 苹果
```

# 过滤器

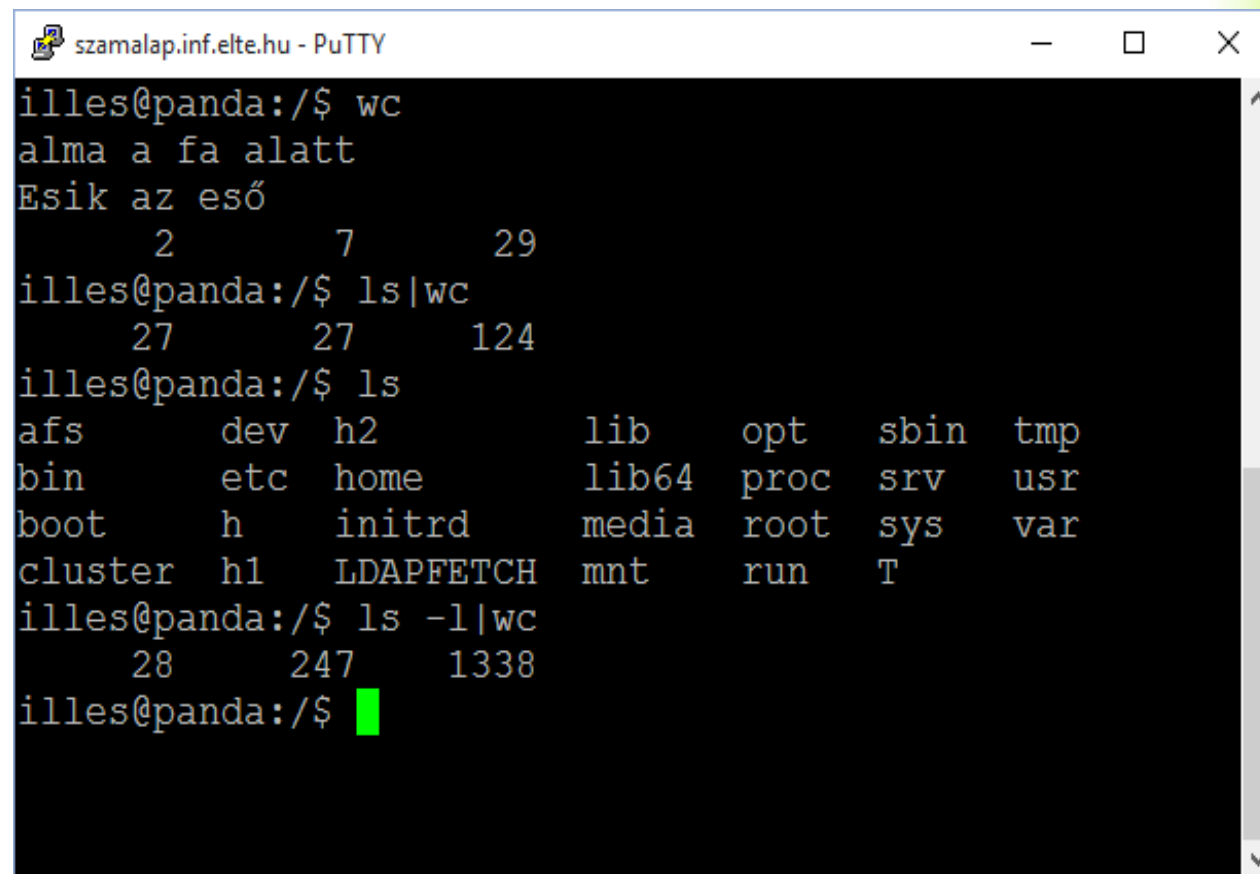
- 命令与过滤器
  - 它的输入可以是其他命令的结果!
- 过滤器本身并不是一个过滤器!
  - 至少它是关于两个命令的连接！"。
- 操作员符号。|
- 这种非常有名的、经常使用的命令是，例如wc!
  - 字数统计!
  - 任务：计算行数、字数、字符数!
    - `wc [-lwc] [file]` #没有参数，它会等待输入数据，直到CTRL+D!

\_\_\_\_\_



# 过滤器的使用

- 可以像普通命令一样使用，没有参数，在这种情况下，它等待来自键盘的数据！
  - 输入结束：CTRL+D
  - 使用它的参数，输入可以被修改！
  - 例如：`wc apple.tree`
- 滤镜一样的，有性格的！



The screenshot shows a PuTTY terminal window titled 'szamalap.inf.elte.hu - PuTTY'. The user 'illes' is at the prompt 'illes@panda:/\$'. They execute the following commands and receive the following outputs:

```
illes@panda:/$ wc
alma a fa alatt
Esik az eső
      2      7      29
illes@panda:/$ ls|wc
      27      27      124
illes@panda:/$ ls
afs      dev  h2      lib      opt      sbin  tmp
bin      etc  home    lib64    proc     srv   usr
boot     h    initrd  media    root     sys   var
cluster  h1   LDAPFET mnt      run      T
illes@panda:/$ ls -l|wc
      28      247      1338
illes@panda:/$
```

The terminal output shows the results of the 'wc' (word count) and 'ls' (list) commands. The 'wc' command shows the number of lines, words, and characters for the file 'alma a fa alatt'. The 'ls' command shows the contents of the root directory, including 'afs', 'bin', 'boot', 'cluster', 'dev', 'etc', 'h', 'h1', 'home', 'initrd', 'LDAPFET', 'lib', 'lib64', 'media', 'mnt', 'opt', 'proc', 'run', 'sbin', 'srv', 'sys', 'tmp', 'usr', 'var', and 'T'. The 'ls -l' command shows the long format listing of the root directory.

# 重要的过滤器

- 重要的，现成的过滤器
  - Tee, tr, cut, sort, uniq, wc, grep等。
  - 鸡蛋。

```
$ who >names  
$ 排序名称 #按行排列  
$ who|sort -r -u # 倒序，唯一线  
$ who|wc-l #登录的用户数
```

- 不要忘了问你的男人(ual)。
  - 例如：人的分类，等等。



# Tee -, tr 命令

- **tee** - 它将传递的内容（通过管道）复制到参数所给的文件中。
  - 例如：`ls -l|tee dir.txt|wc -l, who|tee -a users.txt|sort -r。`
    - `-a`参数：给每个文件添加文本
- **tr-translate**, 它有两个参数，两个字符组。要翻译的字符到达标准输入。如果它们是在第一个字符组中给出的，**tr**将用第二个字符组给出的字符替换它们。
  - 例如：`echo appends|tr pend rive #到达。`

- 与之相似的是sed y命令的用法!(我们以后会看到它!)

修改后：2020年。  
10.08.

## 计算机系统

---

页码：16

# 切割--切出

- 我们可以从标准输入或文件的行中切出字段、列!
- `cut -c1-5` #删去1-5个字符
  - 例如：`date|cut-c4-8` # 10月
- `cut -f1,3,5-7` #砍掉1,3,5,6,7字段
  - 默认的分隔符是:Tab
  - 新的分隔符：`-dchar`
  - 例如：`cat/etc/passwd|cut -f1,7 -d。` # 名字，外壳

修改后：2020年。  
10.08.

# 计算机系统

---

页码：17



# Grep-过滤线

- 选择符合参数所给模式的线条。
- 重要参数。
  - -v 不符合给定模式的线条
  - -i 不区分大小写
  - -w 只选择它作为一个完整的词（comPUter不）。
  - -r 对给定的(参数)目录进行递归。
  - -l 只写出文件名。(在文件内部搜索)。
  - -c 只写出拟合线的数量
  - -n 行的编号。

# grep的用法

- 过滤器的例子。
  - `Cat names|grep Steve` # 结果是，我们得到了包含Steve的行。
- 命令的语法。
  - `grep -r 'Real Madrid' ./script` # 它在目录script的文件中搜索包含Real Madrid的行。
  - `grep -r -l par *` # 在每个文件、每个子目录中，它都会搜索单词par。因此，它只写文件名。

修改后：2020年。

10.08.

## 计算机系统

---

页码：19

# 模式、正则表达式

- 定义一个（文本）模式 - 正则表达式 - 特殊字符
  - ^图案必须从线的起点开始适合。
    - 例如：'^apple'：苹果一词在行首。
  - 美元的图案必须适合于线的末端。
    - 例如：'peach\$'：桃子一词在行尾
  - .任何字符
  - \* 前面的图案重复了0次或更多的次数!
    - 例如：'^apple.\*tree\$' - 苹果和树之间可以是任何数量的字符(0也可以)

修改后：2020年。

10.08.

# 计算机系统

---

页码：20

# 样品一。

- `cat file | grep "^$"#空行`
  - 一个文件中有多少个空行？ `cat file | grep "^$" | wc -l`
- `cat file | grep "^$ FTC.*\ $"# 在开始和结束的位置$ char, 在第一个$ FTC之后, 以及任何char!`
- `cat file | grep "/-"# 任何地方的 /- 字符。`
- `cat file | grep "-/"# 错误, -字符是命令选项的前缀, 没有/选项!`
- `cat file | grep "/-/"# 这是确定的, -/ 字符在任何地方都`

# 可以!

修改后：2020年。  
10.08.

## 计算机系统

---

页码：21

# 其他模式

- 要给字符集。[..]
  - [a-z] (小字)
  - [^a-c] 不是a,b,或c。
  - [A-Za-z0-9] 字母数字型 (数字或字母) 。
    - \w 字母数字, 与前者相同。
    - \W 不是字母数字
  - \d 位数, 与[0-9]相同
  - \s space, tab, line break
- 合适的词语
  - < 单词开头, 例如 : `grep "\<Zol"`
  - \> 词尾, 例如 : `grep "\>tán"`。
  - \b 开始和结束时有空格 Pl: `grep '\bpista\b'` 。



\_\_\_\_\_

## 样本二。

- `cat file | grep [-a] # -或char搜索`
  - `cat file | grep [a-]` #相同。-char可以在第一个或最后  
一个位置!
  - `cat file | grep [a-c]` # the a-c interval, soa orb orc chars
  - `cat file | grep [-ac]` # The -ora orc chars
- `cat file | grep [FTC]{})` # F或T或C和{)之后] 字符
- `cat file | grep [FTC\]{}) ?` # 在[]字符内没有规范的含义，例如：`\d`或。
- `cat file | grep []FTC{)}` # 或 ]或其他字符。

修改后：2020年。  
10.08.

## 计算机系统

---

页码：23

# E(F)grep-pattern fitting I.

- `egrep, fgrep` - 扩展 `grep -E, fixed, grep-F`
  - 意思是说："我们的关系", 也就是关系。
  - 例如：`ls | egrep "par|example"`
  - + 以前的模式至少重复一次
  - 例如：`ls | egrep "\<p+f" #` 在单词的开头1或更多的p, 然后是一个f字母。
  - ?前一个图案重复0或一次。
  - 例如：`ls | egrep "param\d?" #` param, param1, param2, ...
  - {n}前一个字符准时n次!

- 例如：cat appletree|egrep ^[a-b]\w{5}。

修改后：2020年。  
10.08.

## 计算机系统

---

页码：24

# E(F)Grep - 模式拟合II。

- {2,4}前一个图案重复2、3或4次
  - -{1,}以前的模式至少有一次。
- () 从一个图案制作一个组。
  - - 它对于实现重复是很有用的。通常情况下，它的后面是重复的命令+,?,{n}。
  - -E.g: [0-9]{8}(\s[0-9]{8}){1,2}#银行账户
    - 代替[0-9]的d也不错。
- 特殊字符是实用的，可以在前面加上一个字符。
  - -例如。 ^[+-]? \d+ ([,\.] \d+)? #一个带有小数点或逗号的有符号的数字

- 更多的例子，请看男人！。

修改后：2020年。  
10.08.

计算机系统

---

页码：25

# 其他模式

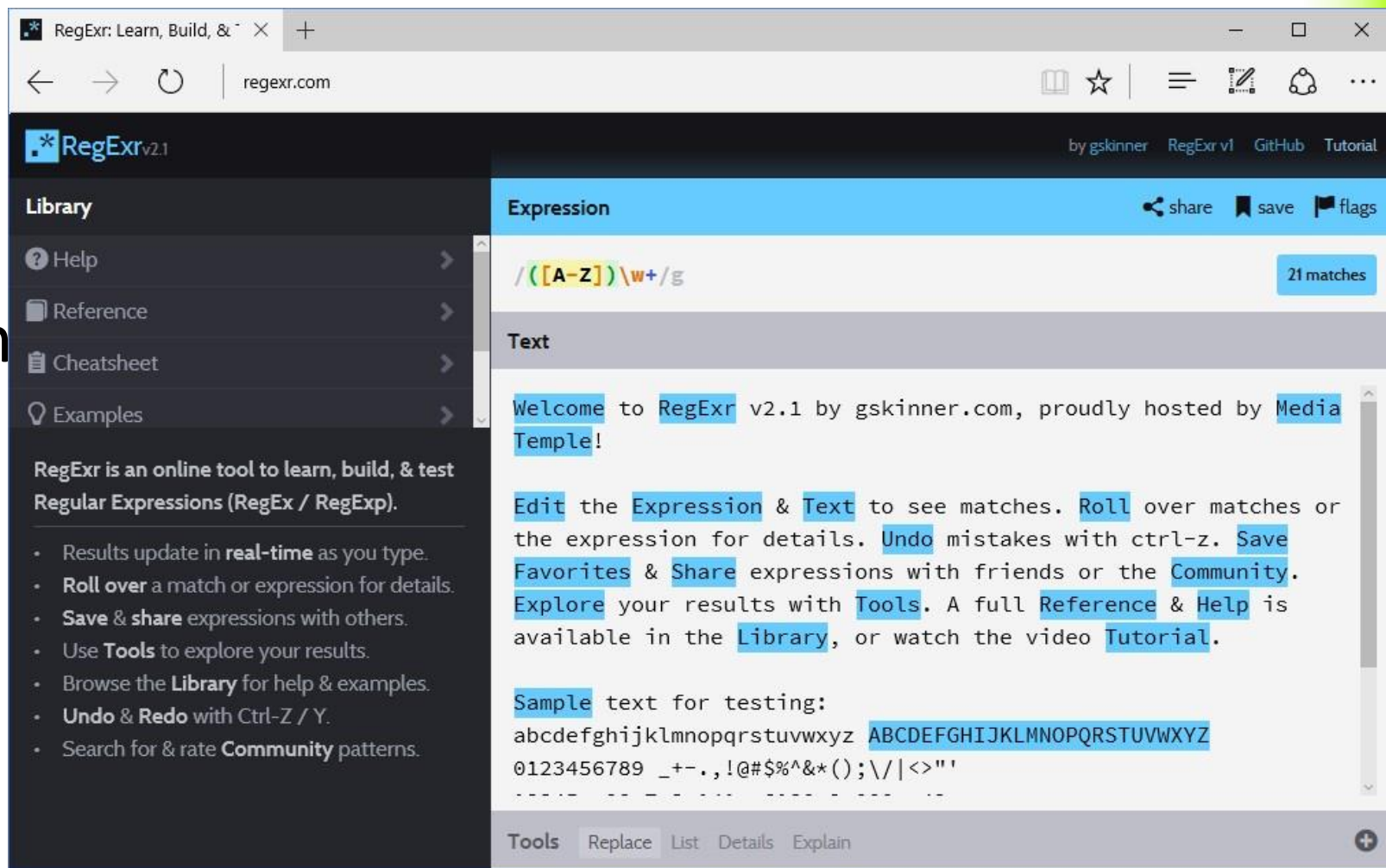
- 电子邮件地址（简称）。
  - `\w+[\.-_\w*]*@\w+(\.\w+)+`
- 时:分:秒
  - `[01][0-9]|2[0-3]:[0-5][0-9]:[0-5][0-9]`
- 等等。
- 每种编程语言都包含这个或它的扩展版本!





# 正则表达式 - 在线

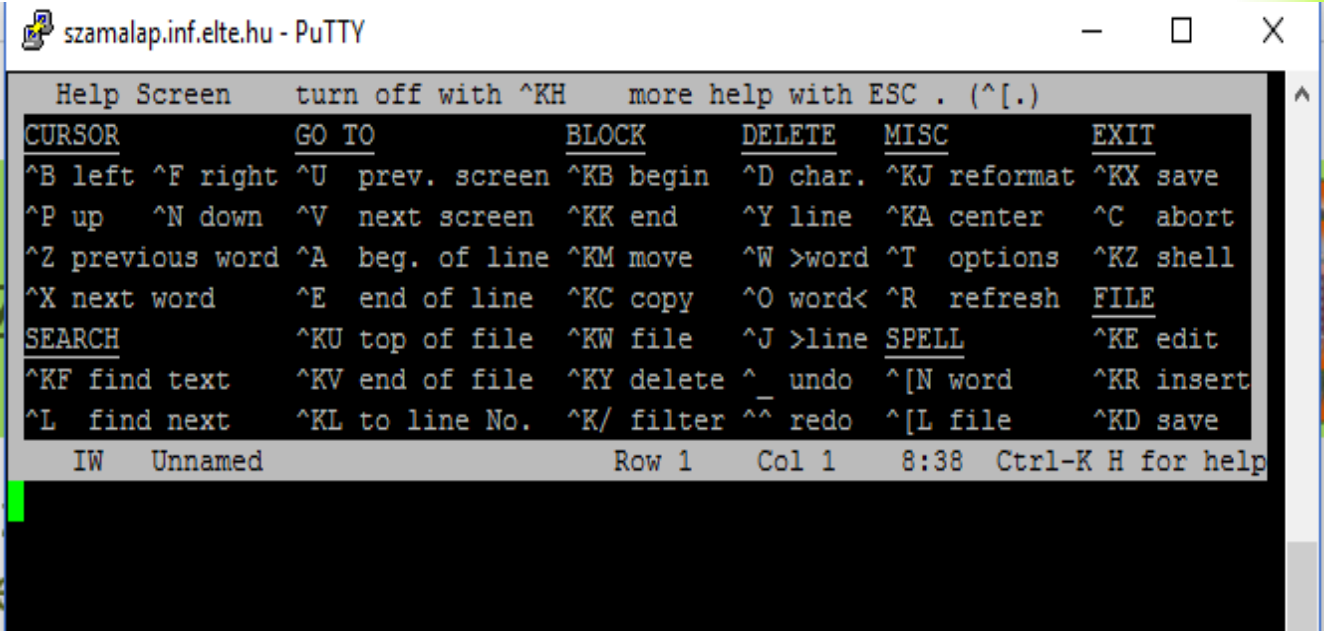
- 你可以找到在线网站
- <http://regexr.com>
- <http://regex101.com>
- 等等。





# 文本编辑器 - I.

- 有广泛的编辑可供选择!
- 典型的有：vi、pico、mcedit、joe等。
- 在显示屏上可以看到对基础使用的帮助!
- 除了vi!
- 你可以在图片上看到JOE的帮助!



# 文本编辑器（VI或VIM） - II.

- 一些vi(m)功能!
  - 两种工作模式，命令或编辑模式!
  - 从命令模式切换到编辑模式：i、a、o，等等。
  - 要从编辑器切换到命令模式。  
ESC
  - 在命令模式下：保存：w名称，保存和退出：wq（退出）。
  - :帮助(见图)
  - 从帮助屏幕退出：q
  - 不保存就退出：q!

修改后：2020年。  
10.08.

```

szamalap.inf.elte.hu - PuTTY
help.txt*      For Vim version 7.4.  Last change: 2012 Dec 06

                VIM - main help file

                k
Move around:   Use the cursor keys, or "h" to go left,      h  l
               "j" to go down, "k" to go up, "l" to go right.  j
Close this window: Use ":q<Enter>".
Get out of Vim:  Use ":qa!<Enter>" (careful, all changes are lost!).

Jump to a subject: Position the cursor on a tag (e.g. |bars|) and hit CTRL-].
With the mouse:   ":set mouse=a" to enable the mouse (in xterm or GUI).
                  Double-click the left mouse button on a tag, e.g. |bars|.
Jump back:        Type CTRL-T or CTRL-O (repeat to go further back).

Get specific help: It is possible to go directly to whatever you want help
                   on, by giving an argument to the |:help| command.
                   It is possible to further specify the context:
                                     *help-context*
                   WHAT                PREPEND    EXAMPLE
                   Normal mode command (nothing)  :help x
help.txt [Help] [RO]                                     1,1      Top
[No Name] [+]                                           1,0-1    Top
"help.txt" [readonly] 221L, 8249C
```

# VI(M) - 无名-有名缓冲区

- 剪切 - 复制 - 粘贴

- 切：`dd`，删除(切)实际行，内容将在未命名的缓冲区中。
- 复制：`yy`，将实际行复制到未命名的缓冲区内
- 粘贴：`p`，复制（插入）缓冲区的内容
- 切割，复制多行。`5dd`或`4yy`，剪切5行，复制4个4行到缓冲区。

- 命名的缓冲区

- 我们只能使用一个字符的长缓冲区名称，在双引号之后：例如，`"a`
- `"s2yy #` 将实际的2行复制到S缓冲区中。



- "sp # 粘贴S缓冲区的内容

2020.10.08.

---

纪念活动

30

