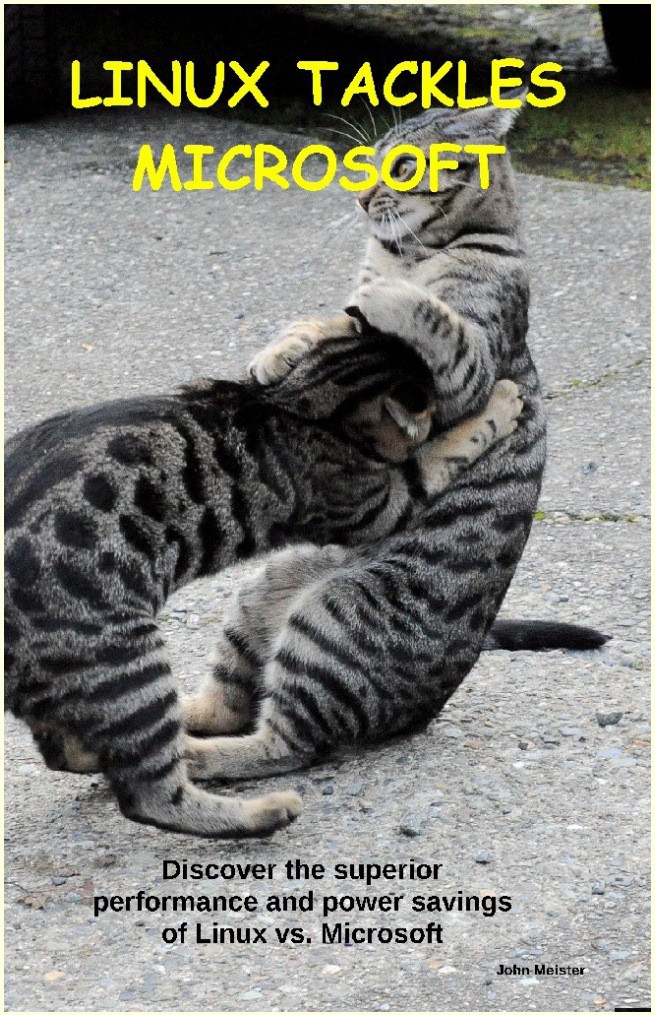
Get a CLUE* with Linux:

**command line user environment*

Get a CLUE with Linux;

Or, how to save 182 manhours
analyzing test results by using
LINUX regular expression tools
at the command line
rather than in Microsoft Excel.

(spreadsheets are amazing tools!)



Get a CLUE* with Linux -

*Command Line User Environment

- Regular Expressions in Linux **saved around 182 Manhours** during the remaining **14 tests** of a 19 test cycle with 10 reports for each cycle.
- **Linux saved about 13 hours per test cycle of sorting and comparing in Excel** by using regular expression commands in a Linux shell, taking less than 15 minutes to type the commands.
- A BASH shell script was written to automate the process; it runs in less than **FIVE SECONDS.**
- **13 hours to 15 minutes to less than 5 seconds for each cycle.**
- *First an overview of Linux; then the script will be explained.*

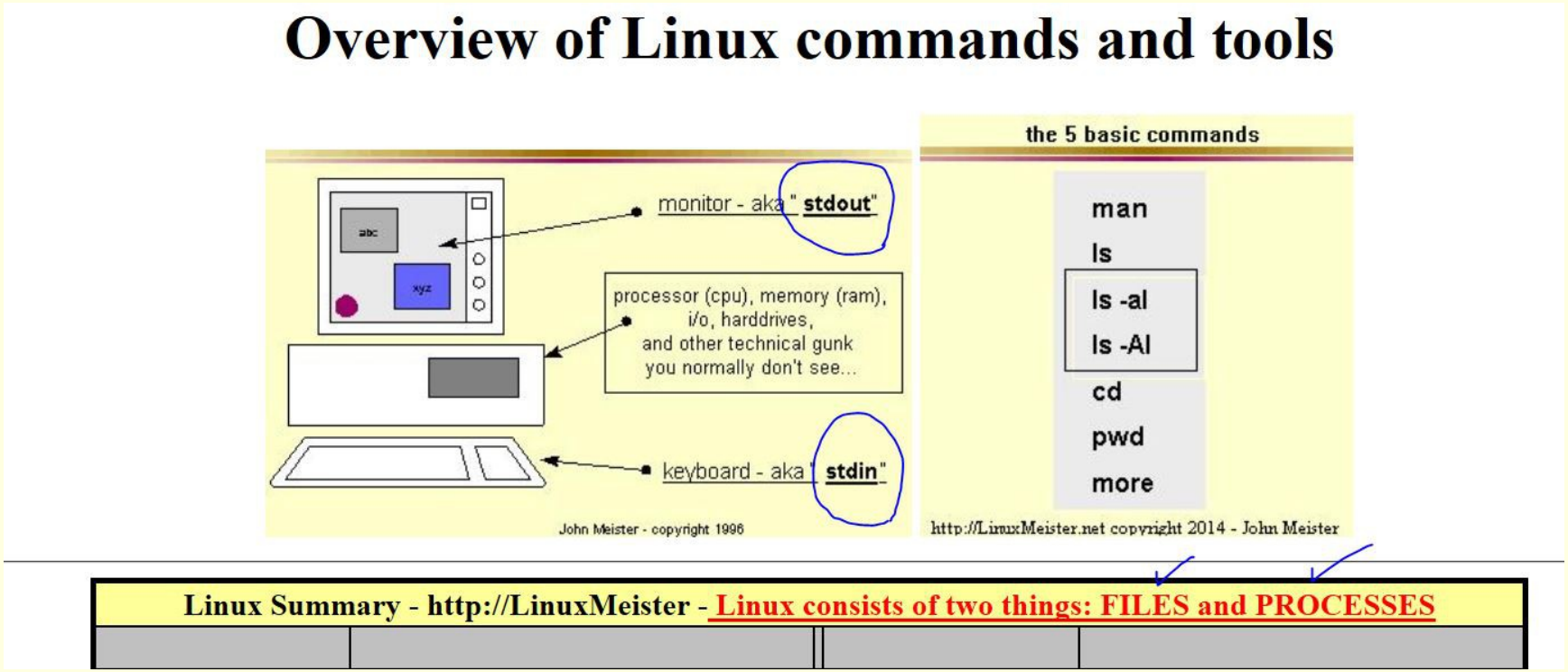
Linux consists of TWO THINGS:

Files and Processes

***Files** are used to create **processes** and*
***processes** do things with or to **files**.*

Files may be created or modified using **"stdin"** - aka, keyboard or other devices,
and also from the output of processes, known as **"stdout".**
There is a third value, **"stderr"**, which reports errors to **"stdout"**.

stdin, stdout and stderr can all be redirected, piped, filtered and modified at the command line.
Linux uses **INTERPROCESS COMMUNICATION** - ("piping") allows the output from one process to be the input to another.



Linux Summary - http://LinuxMeister.net - *Linux consists of two things:* **FILES and PROCESSES**			
Linux Overview:	http://LinuxMeister.net/overview-LINUX.pdf	**5 commands:**	man, ls, cd, pwd, more
Distributions:	http://distrowatch.com	manual (RTM):	**man**
SuSE	http://opensuse.org	list files:	**ls**
Debian/Ubuntu	http://linuxmint.com/		**ls -al**
Centos/RedHat	http://www.centos.org/		**ls -Al**
Browser	http://mozilla.org/firefox/		*ls [-aAlRst...]*
Email	http://mozilla.org/thunderbird/	change directory:	**cd**
Office Suite	http://openoffice.org		cd [path]
Office Suite	http://libreoffice.org	present directory:	**pwd**
Image Editing	http://gimp.org	**in Linux, less is...**	**more**
Linux consists of:	**FILES and PROCESSES**	Shells: sh, bash, ksh	**history / script**
special characters:	**& ; | * ? ' " ` [ ] ( ) $ < > { } ^ # / \ % ! ~ -**	**user config:**	.bashrc
process control	**ps / kill / nice / jobs / bg / fg**	**shell functions**	export / env / echo
stdin, stdout, stderr	*redirects:* **< >** *pipe:* **|** *comment:* **#**	**shell aliases:**	ll="ls -al"
vi Editor:	**vi <filename>**	**File Management:**	**chown / chmod**
hidden cursor keys:	**h,j,k,l**		**cp (-rp)**
move left	**h**		**locate**
move down	**j**		**lsof**
move up	**k**		**mv**
more right	**l**		**rm (-rf)**
command mode	**<esc> <command>**		**touch**
edit/text mode	**i, I, a, A, o, O…<start typing>**		**tree**
cmd: beginning of line	**0**		**whe**
cmd: end of line	**$**		**wh**
cmd: delete	x, dd (line), D or d$ (to end of line)	**File systems:**	**du / d**
to save:	**<esc> :w**		**fdisk [-l]**
to save as new file:	**<esc> :w <filename>**		**fs**
to exit w/out saving:	**<esc> :q!**	physical volume (LVM)	**pvcr**
to save and exit:	**<esc> :wq**	volume group (LVM)	**vgcr**
to exit forcing save:	**<esc> :wq!**	logical volume (LVM)	**lvcr**
			mi
Regular Expressions	**awk / grep / sed**		**pv/vg/lv**
tools and filters:	**head / tail**		**mount /**
	find / tee / nl / xargs		
	tee [-a] / echo	Network:	**ifconfig [-**
	tr / col -b / expand		**route / ip /**
	sort / uniq / fmt		**ping / tra**
	cut / paste / join / split		**hostnam**
	diff / cmp	Pkg Management:	**rpm / dpk**
	wc [-l]		**tar / cpio /**
			yum /
SSH	**authorized_keys**	Sys Admin	**adduser**
	known_hosts		**dmesg**
	scp (-rp)		**sudo**
	ssh [-l <user>] hostname (ip)		**w / last /w**
	ssh-keygen		**top / vmsta**

This presentation:
http://linuxmeister.net/Scripts/Engineering/

One Page LINUX Overview:
http://linuxmeister.net/Overview/

a quick overview to view later:
http://LinuxMeister.net/Overview/Linux-PowerPoint-2004-overview.pdf

RESOURCES potentially available at work:

1) **Cygwin in Windows – internal sources**
2) **BASH in Win10 – upon approval**
3) **Linux in a VM – upon request/approval**
4) **Linux on a system – upon request/approval**
5) **Linux accessed via existing system in datacenter.**

Linux Overview:	Linux Overview PowerPoint from 2004
Distributions:	DistroWatch.com (250+ options!)
SuSE	OpenSuSE.org (SuSE)
Debian/Ubuntu	LinuxMint.com (Debian/Ubuntu)
Centos/RedHat	CentOS.org (RedHat supported)
Browser	Firefox - Mozilla (Netscape)
Email	Thunderbird
Office Suite	OpenOffice.org
Office Suite	LibreOffice.org
Image Editing	darktable.org (lightroom)

The LINUX (and UNIX) Philosophy

The philosophy is a result of more than twenty years of software development and has grown from the UNIX community instead of being enforced upon it. It is a defacto-style of software development. The nine major tenets of the UNIX Philosophy are:

1. small is beautiful
2. make each program do one thing well
3. build a prototype as soon as possible
4. choose portability over efficiency
5. store numerical data in flat files
6. use software leverage to your advantage
7. use shell scripts to increase leverage and portability
8. avoid captive user interfaces
9. make every program a filter

The Ten Lesser Tenets

1. allow the user to tailor the environment
2. make operating system kernels small and lightweight
3. use lower case and keep it short
4. save trees
5. silence is golden
6. think parallel
7. the sum of the parts if greater than the whole
8. look for the ninety percent solution
9. worse is better
10. think hierarchically

FROM THE BOOKS:

The Unix Philosophy -- Mike Gancarz
Linux and the Unix Philosophy -- Mike Gancarz, Jon "Maddog" Hall

---

To learn Linux, start with these 5 commands:

man, ls, cd, pwd, more

To compare Microsoft DOS commands with Linux see these pages:

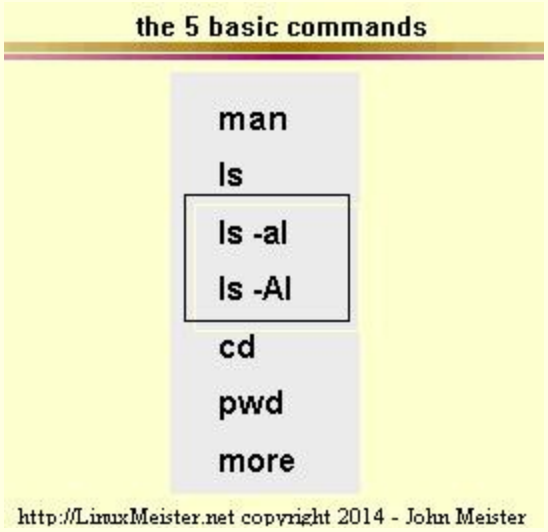
http://linuxmeister.net/Microsoft/DOS-Linux-cmds.html

http://linuxmeister.net/Microsoft/PowerShell-vs-BASH-commands-compared.html

http://linuxmeister.net/Microsoft/LOADTEST-OS-COMPARISON.html

In Linux, we have many ways to show the contents of a file, **more** is recommended because it filters, see also: "less", "**cat**", "tac" and other commands (*use* **apropos** *to find other cmds, and* **man** *to learn about them*)

5 commands:	man, ls, cd, pwd, more
manual (RTM):	**man** - help pages, command options
list files:	**ls**
	ls -al
	ls -Al
	ls [-aAlRst...]
change directory:	**cd**
	cd [path]
present directory:	**pwd**
in Linux, less is...	**more**

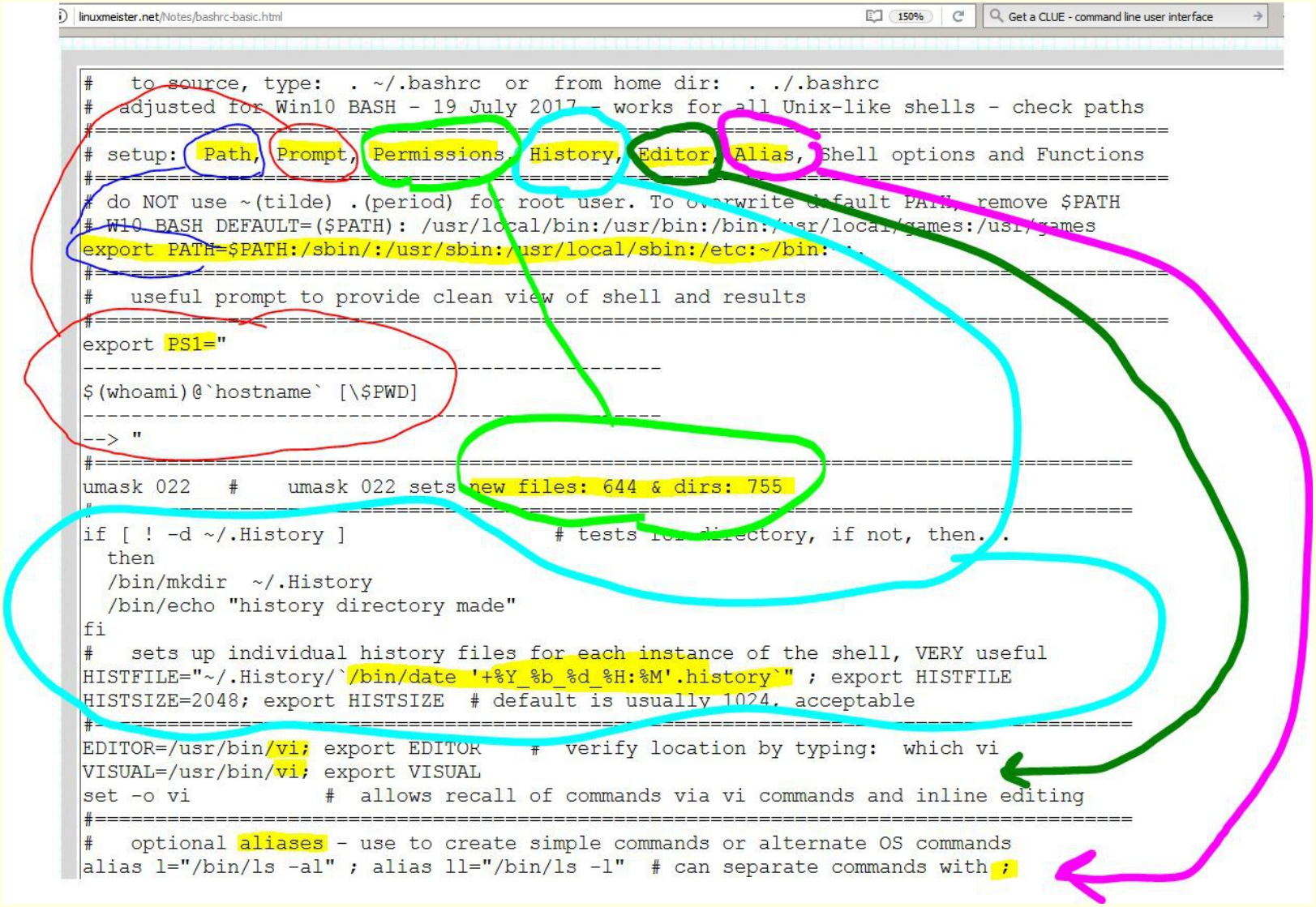


Setting up your SHELL environment will help productivity, visibility and create custom tools:

see also #1 of the lesser tenets: **allow the user to tailor the environment...**

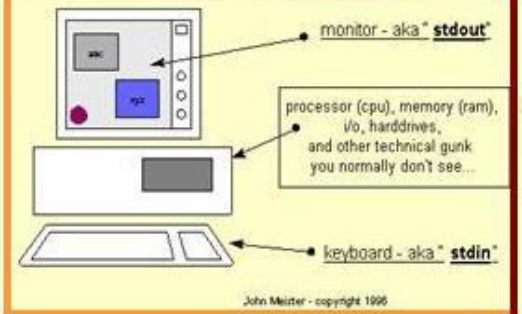
http://linuxmeister.net/Intro-to-Linux/bashrc-n-history-details.html

http://linuxmeister.net/Notes/bashrc-basic.html



Files, filesystems and terminology

a starting point



UNIX Files

- Plain — Standard files
- Special — Block & Device
- Directory — Directory

there are others, e.g. links, FIFO, socket, pipes... see man file

John Meister - copyright 1996

The Essential UNIX Commands

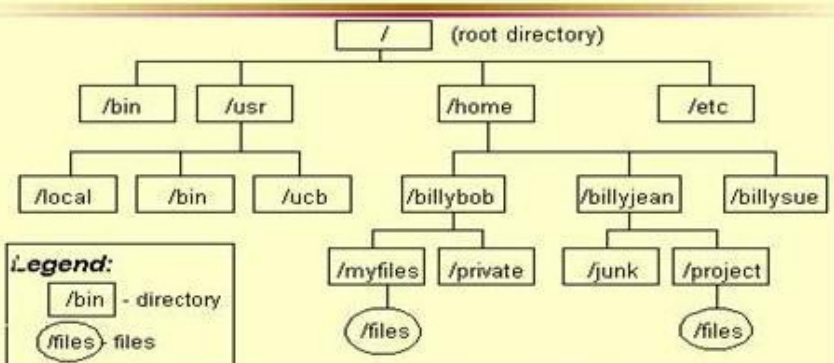
man
ls
ls -al
ls -Al
cd
pwd
more

John Meister - copyright 1996

Linux Vocabulary (YMMV)

#	= pound, hash - (comment)
!	= bang - (a "force" to save, or "not" in queries)
*	= splat, star - (wildcard)
vi	= "v" "i" - (**the** editor – you must learn it!)
/	= slash, root (root filesystem)
/etc	= (slash) "et cee" (configuration)
/bin	= (slash) bin (executables)
/usr	= (slash) "user"
/opt	= (slash) "opt" (optional s/w)
/sbin	= (slash) "s bin" (admin tools)
\	= back slash (MicroSoft slash)

UNIX Filesystem Tree



John Meister - copyright 1996

john meister - 2013

You need to be aware of "key words", built-ins and commands, as well as **special characters.**

SPECIAL CHARACTERS

& ; | * ? ' " ` [ ] ( ) $ < > { } ^ # / \ % ! ~ -

(spaces have significance)
Be careful with the use of these characters at the command line or in file names. If you have unexpected results with a command or a script it may be related to one of these.

&	- used to place jobs in the background, *ex: mozilla &*
;	- separates commands, *ex: ls ; cd / ; ls*
|	- used to "pipe" processes, *ex: cat filename |col -b > newfile*
*	- wildcard, one or more characters
?	- wildcard, ONE character only
' " `	- various uses within shells and scripts (single quote, quote, and grave)
[ ] ( )	- used to pass variables and define ranges
$	- represents a process or a variable, *ex: echo $SHELL*
< >	- redirection
{ }	- used to pass parameters, see find examples later in course
^	- up caret - often used to define keys, *ex: stty erase ^H (sets backspace)*
#	- pound, used at the front of a comment, ignored by shell and scripts
/	- path or escape sequences
\	- escape special meaning in command, e.g. "\rm" - disables the alias "rm -i"
%	- used by sed as a parameter, see later examples of use in vi and sed
!	- bang, used to escape a shell or vi, also combined with pound to instruct "command interpreter", *ex: #!/bin/sh*
~	- tilde, used to indicate home directory on many systems, other uses
-	- dash, used to provide switch information to commands

Sorting things out:

naming files and directories (**NEVER** use spaces *or most special characters*)

The basics of Computers

Computers sort by Name, Date or Size.

When sorted by name a computer will sort **numeric, then alphabetic.**

To create easily sorted DIRECTORIES (folders) it is often useful to use leading NUMBERS, however, a zero placeholder is needed to sort numbers above 10, e.g. 0-15 would require numbers below 10 to have a leading zero, otherwise it would not sort correctly. **A true sort shows 20 listed before 2.**

EXAMPLE:

```
--> cat dirnames.txt | sort
002_directory
02_directory
10_directory
11_directory
12_directory
1_directory
20_directory
2_directory
3_directory
4_directory
5_directory
6_directory
7_directory
8_directory
9_directory
a_directory
B_directory
C01_directory
C10_directory
C1_directory
C_directory
filelist.txt
```

correctly sorted, will work on ALL computers:

```
001_directory
002_directory
003_directory
004_directory
005_directory
006_directory
007_directory
008_directory
009_directory
010_directory
011_directory
012_directory
020_directory
a_directory
B_directory
C01_directory
C10_directory
C1_directory
C_directory
filelist.txt
```

Win7 appears to "understand" what you're trying to do… but if you move this list to another application or system it won't sort the same… and cause a lot of headaches and rework.

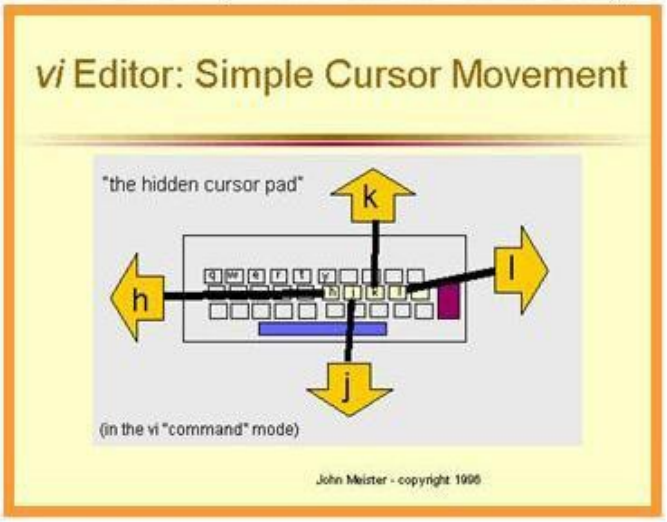
These convenience features often create more problems than they solve…

Users must be aware of how things work to minimize challenging problems.

the best editor in the world: vi

to enable at the command line: **set -o vi**

The vi editor (command mode - basics):



The vi editor (command mode - basics):

KEY	EFFECT – CURSOR MOVEMENT
w	Next beginning of word, treating punctuation as new word
b	Previous beginning of word, treating punctuation as new word

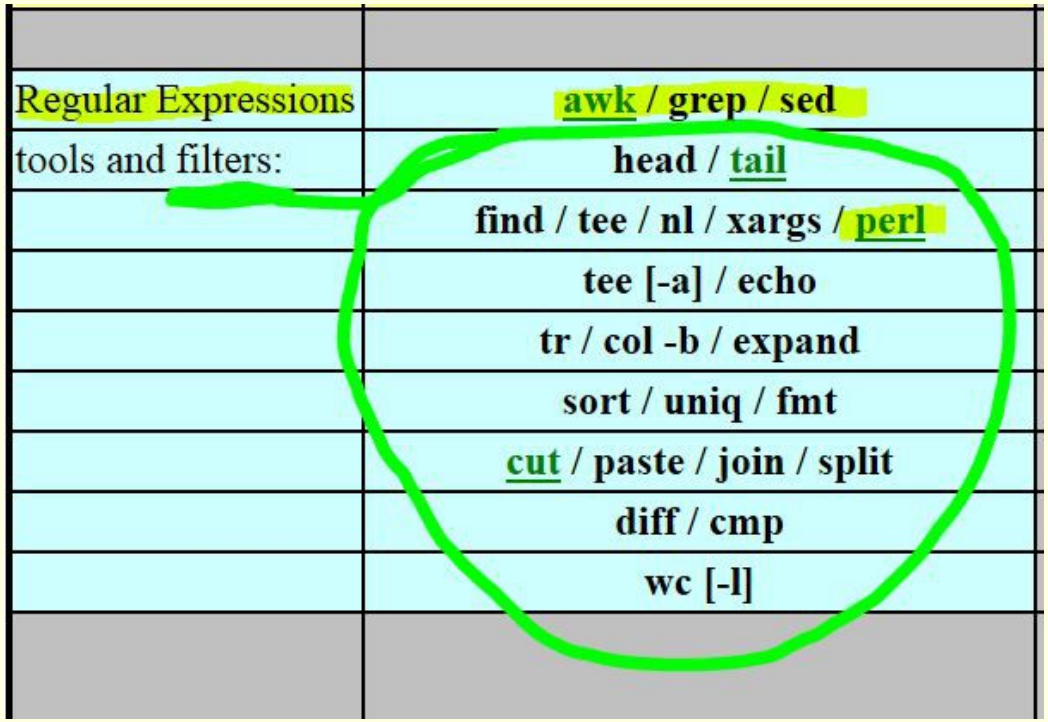
KEY	EFFECT – SCREEN MOVEMENT
^f	Scroll one screen forward
^b	Scroll one screen back

The vi editor (command mode - basics):

KEY	EFFECT
h	Move one character **left**
j	Move **down** one line
k	Move **up** one line
l	Move one character **right**
0	Move to **beginning** of current line *(Note: this is "zero" key)*
$	Move to **end** of current line
i	Insert text
o	Insert line below cursor
A	Append at end of line
esc	Command mode
:	Invoke "ex" command
r	Replace character
cw	Change word
x	Delete character
dw	Delete word
dd	Delete line
p	Put (paste) contents of buffer
yw	Yank (copy) word
yy	Yank (copy) line
u	Undo last command
.	Repeat last command
U	Undo all changes to line
d$	Delete to end of line
C	Change text to end of line
J	Join lines

Regular Expressions

- awk – **Aho**, *Alfred V.;* **Weinberger**, *Peter J.; and* **Kernighan** *Brian W. (1977)*
  https://en.wikipedia.org/wiki/AWK#Versions_and_implementations
  http://linuxmeister.net/Commands/awk-example.html http://linuxmeister.net/Notes/tabs-convert-to-spaces.html
- sed – *stream editor (also used with Perl)*
  http://linuxmeister.net/Notes/using-sed-to-count-executable-files.html
  ls `env | grep PATH | *grep -v XNLSPATH* | sed -e 's/^PATH=//' | sed -e 's/\:/ /g'` **2>/dev/null** | wc -l
- grep – *global regular expression search*
  http://linuxmeister.net/Notes/using-find-grep-xargs.html http://linuxmeister.net/Commands/use-of-grep.html
  find . -type f -name *.html | xargs grep **johnmeister.com/linux** | perl -pi -e 's$http://**johnmeister.com/linux**$http://**linuxmeister.net**$g'



SORTING TEST RESULTS USING REGULAR EXPRESSIONS

- A computer system under test in the early development stages monitored an ARINC bus and reported statuses. Some of the notifications were considered nuisance messages because of intial settings.

  Some errors might be related to a sensor with too narrow a hysteresis, a parameter incorrectly set in the s/w, an out of adjustment gadget, or less likely, an actual defective component.

  Analyzing his process and test results, we determined the plain text files had a consistent format with an easily spotted "key field", i.e. the ATA Maintenance Chapter.

  The reports had names unique to the product under test. This info was also in each report.

  To protect the original reports and to make handling the sorts easier,
  We copied the originals with new names from 01.txt to 10.txt, with a "txt" suffix for interoperability.
- *note: the content of these reports and the specific systems involved are not included, "dummy data is used"*

- 10 reports were generated for each test cycle of the system
- 19 test runs were made under different conditions to ensure full compliance
- a USB device is used to copy the text files from the system
- The 10 reports were copied into Excel
- each report would be sorted in Excel by ATA chapter
- copying and sorting the reports would take an hour
- after all 10 reports were sorted, the engineer compared them, looking for common errors
- It took another 3 hours to compare those 10 sorted reports
- The engineer then examined each of the common errors in the original reports
- Once he determined the issue, he'd resolve that one and move to the next
- The engineer had 13 hours invested in this analysis for just ONE test cycle, there were 19 total tests but he didn't come to the lab until he had completed 5 test cycles, leaving 14 cycles for the savings.
- Had we run the script for all 19 cycles the savings could have been **247 manhours.**

https://havrel.honeywell.com/docs/inforesources/ATA_Chapter_Descriptions.PDF

ATA Chapter Descriptions

By System (Alpha A,B,C)		By Code (Numerically 1,2,3)	
CH	System	CH	System
83	Accessory Gear Boxes	11	Placards
21	Air Conditioning	21	Air Conditioning
49	Airborne Auxiliary Power	22	Autopilot
22	Autopilot	23	Communications
91	Charts	24	Electric Power
23	Communications	25	Equipment & Furnishing
52	Doors	26	Fire Protection
24	Electric Power	27	Flight Controls
72	Engine	28	Fuel
75	Engine Air	29	Hydraulic Power
76	Engine Controls	30	Ice & Rain Protection
78	Engine Exhaust	31	Instruments
73	Engine Fuel & Control	32	Landing Gear
74	Engine Ignition	33	Lights
77	Engine Indicating	34	Navigation
79	Engine Oil	35	Oxygen
72	Engine Reciprocating	36	Pneumatic
80	Engine Starting	37	Vacuum
72	Engine Turbine	38	Water/Waste
82	Engine Water Injection	49	Airborne Auxiliary Power
25	Equipment & Furnishing	51	Structures
26	Fire Protection	52	Doors
27	Flight Controls	53	Fuselage
28	Fuel	54	Nacelles/Pylons
53	Fuselage	55	Stabilizers
29	Hydraulic Power	56	Windows
30	Ice & Rain Protection	57	Wings
31	Instruments	60	Std. Practices-Props
32	Landing Gear	61	Propellers
33	Lights	70	Standard Practices Engine
54	Nacelles/Pylons	71	Power Plant-General
34	Navigation	72	Engine
35	Oxygen	72	Engine Turbine
11	Placards	72	Engine Reciprocating
36	Pneumatic	73	Engine Fuel & Control
71	Power Plant-General	74	Engine Ignition
61	Propellers	75	Engine Air
55	Stabilizers	76	Engine Controls
70	Standard Practices Engine	77	Engine Indicating
60	Std. Practices-Props	78	Engine Exhaust
51	Structures	79	Engine Oil
81	Turbines	80	Engine Starting
37	Vacuum	81	Turbines
38	Water/Waste	82	Engine Water Injection
56	Windows	83	Accessary Gear Boxes
57	Wings	91	Charts

basic info found in the reports and scoping the file sizes

1) Determine how many lines of data were in the 10 files.
2) Copy originals as **01.txt to 10.txt**, so we can use "?" wildcards.
3) "cat" all 10, then count lines using wc.

--> cat ??.txt | wc -l

41025

that's 41,025 lines of information.

4) add more filters to see how many possible Messages there could be:

-> cat ??.txt | grep Maint | grep - | wc -l

5640

That's 5,640 messages included in that batch of 10 reports.

5) Now we needed to find out which were common across all 10 reports and work those.
6) Technical content for each of the reported errors ranged from 14 to about 23 lines long,

using **"grep" with the -A for "after" the pattern,** we could create a report that gave most of the details.

Initially the engineer did the steps manually. A script has been written to automate the steps.
What took 13 hours per test run in EXCEL now takes less than 5 seconds in a Linux shell..

From 13 hours... to less than 5 seconds.

Next, find one error message in these 10 reports. *The information is obsfucated for security.*

grep to find the pattern 21-18891 and to **display 23 lines after** that.

The goal of course was to eliminate all errors, but initially they focused on the common ones first.

the manual process (15 minutes)

We tested this process with three reports.

--> cat 01.txt | grep Mainten | grep - | awk '{print $4}' | sort > 01-rpt

The awk command will find patterns between delimiters. A space is a default delimiter.
In the string above we are printing the 4th field, the string we found with grep looked like this:

[] Maintenance Message: 21-18891 ACTIVE

The first field is the "brackets", second "Maintenance", third " Message:", and fourth, our numbers.
NOTE: at this point the three files were created with Maintenance codes only listed

```
# -rw-r--r-- 1 luser users 486 Sep 19 14:17 1-rpt
# -rw-r--r-- 1 luser users 414 Sep 19 14:18 2-rpt
# -rw-r--r-- 1 luser users 1143 Sep 19 14:18 3-rpt
```

Once the 3 files were intially used to create the Maint code list test report, then we used the "**compare**" command to compare the 3 (or more) reports: (**comm - compare two sorted files line by line**)

--> comm -12 1-rpt 2-rpt | comm -12 - 3-rpt > common-messages.txt

Then we look at our common messages in the file...
in the example below we saw 7 common messages across all three of the reports. # NOTE: testing output:

--> cat common-messages.txt

```
# 24-19405
# 30-32702
# 30-32703   ...
#
```

Then we used grep (global regular expression - print lines matching a pattern) to find WHICH of the filtered files had those codes...we used grep on each of the original files to find the reports which lead us back to the messages. TESTING:

--> grep 24-19405 *

```
# 1-rpt:24-19405
# 2-rpt:24-19405
# 3-rpt:24-19405
# common-messages.txt:24-19405
# XX_YY_B-7839_2016-07-28_21.38.57Z.txt: [] Maintenance Message: 24-19405 ACTIVE
# XX_YY_AP432A_2016-07-28_21.06.45Z.txt: [] Maintenance Message: 24-19405 ACTIVE
# XX_SPTEST465_2016-05-18_07.46.58Z.txt: [] Maintenance Message: 24-19405 ACTIVE
# ----------------------------------------------
```

Use **grep** to globally find the regular expression "*21-18891*" and then display **23** lines **A**fter that in file 01.txt

--> **grep -A 23 *21-18891* 01.txt**

 # (*NOTE*: *could redirect to a file, or pipe to another process…)*

```
[] Maintenance Message: 21-18891                              ACTIVE
        Left ECS Card has no output on  ECS ARINC 429 Bus


     EF_010203040_longtimeago-01-08_21.28.48Z.txt                    090807060
     TEST DB:   ABC-D-EF41-000D                                PAGE:       7
     MaintComp OPS: HIJ-K-LM32-600J                                DATE: 08JAN18
     OTHER OPS:  Compatible                                     TIME:   2129z

     EXISTING FAULTS All Existing Maintenance Messages
                                    This data is from the Left MaintComp

     21 Environmental Confusion System                                  (29)

        Detected By:
           ARINC Signal Gateway (left), LSCF
           ARINC Signal Gateway (right), LSCF

        Recommended Maintenance Action:
        See Fault Isolation procedure. When using a maintenance
        laptop to show MaintComp data, select FAULT ISOLATION DESTRUCTION button
        below.

     21 Cabin Temperature Confusion System                               (5)
```

Test Result Analysis script – find common messages

```
#!/bin/bash
#  jm - 25jan2018 - ATA info extract from test reports
#################################################################################
for x in `ls ??.txt`    #  for each report find the ATA chapters listed
   do
    Y=`echo $x | cut -c 1-2`
     ### NOTE:  drop the suffix: .txt - cut first two characters to use for new filename
 cat $x | grep Mainten | grep - | awk '{print $4}' | sort | uniq | grep -v ^$   > $Y-rpt
     ### NOTE: grep for key term, qualify with "-", print values in field 4, sort, drop dups and blank lines
   done
#################################################################################
 comm -12 01-rpt 02-rpt | comm -12 - 03-rpt | comm -12 - 04-rpt | \
comm -12 - 05-rpt | comm -12 - 06-rpt | comm -12 - 07-rpt | \
comm -12 - 08-rpt | comm -12 - 09-rpt | comm -12 - 10-rpt > common-messages.txt
#################################################################################
 for z in `cat common-messages`
   do
   grep $z ??.txt            >>      status-ATA-designator-`date +%d%b%y`.txt
   echo "==========="        >>      status-ATA-designator-`date +%d%b%y`.txt
   grep -A 15 $z ??.txt      >> DETAILS-ATA-designator-`date +%d%b%y`.txt
   echo "==========="        >> DETAILS-ATA-designator-`date +%d%b%y`.txt
   done
#################################################################################
##  first grep gets the line with the ATA chapter, 2nd grep read 15 lines AFTER - to see details
##  the echo inserts a divider between messages for reading clarity
#################################################################################
```

...and executing the script...

--> time sh ./get-ATA.sh

====== finding common ATA chapter messages in 10 reports for further analysis ====
Thu, Jan 25, 2018 **3:50:01 PM**
= completed, see ATA...msg-status and ATA...DETAILS for analysis areas =====
Thu, Jan 25, 2018 **3:50:06 PM**
==========

real 0m5.110s *# note: cygwin on win 7 T5810 desktop*
user 0m0.512s
sys 0m3.060s

AFTER RUNNING THE SCRIPT:

---
luser@Luser-Lab [/home/luser/LINUX_REG_EXP_TOOLS/FIND-COMMON_ATA_MSG]

--> ll
total 4317

```
-rwxrwx---+ 1 luser local Lusers 185208 Jan 25 09:03 01.txt
-rw-rw-r--+ 1 luser local Lusers   2466 Jan 25 15:50 01-rpt
... .
-rwxrwx---+ 1 luser local Lusers 441026 Jan 25 09:03 09.txt
-rw-rw-r--+ 1 luser local Lusers   5180 Jan 25 15:50 10-rpt
-rw-rw-r--+ 1 luser local Lusers 104801 Jan 25 15:50 DETAILS-ATA-designator-25Jan18.txt
-rw-rw-r--+ 1 luser local Lusers  11042 Jan 25 15:50 status-ATA-designator-25Jan18.txt
-rw-rw-r--+ 1 luser local Lusers     99 Jan 25 15:50 common-messages.txt
-rw-rw-r--+ 1 luser local Lusers   2513 Jan 25 15:49 get-ATA.sh
```

01.txt:	[] Maintenance Message: 38-12041	LATCHED
02.txt:	[] Maintenance Message: 38-12041	ACTIVE
03.txt:	[] Maintenance Message: 38-12041	LATCHED

….

...performance of script... compare OS & HW

-> **time** sh ./get-ATA.sh

	Env	OS	hardware	min	seconds	processor(s)	speed	memory
1	BASH	**SuSE** Linux 42.1	Dell M4800	0	**0.202**	quad core	2.9 GHz	16 Gb
2	BASH	Ubuntu Linux 14.04	Dell T5810	0	**0.219**	4	3.7GHz	16 Gb
3	BASH	Centos7 Linux	Dell 7520	0	**0.263**	quad core	2.9 GHz	16 Gb
4	**VM/Mac**	**Mint** Mate 17.3	MacBook Air 13" 2012	0	**0.227**	1	2GHz	8Gb
5	BASH	**SuSE** Linux 42.2	Toshiba Portege R600	0	**0.478**	core2duo	1.4 GHz	4 Gb
6	BASH	WINDOWS 10	Dell T5810	0	1.689	4	3.7GHz	16 Gb
7	BASH	MacOSX 10.13.3	MacBook Air 13" 2012	0	1.941	1	2GHz	8Gb
8	**CygWin**	Windows 7	Dell T5810	0	5.110	4	3.7GHz	16 Gb
9	**CygWin**	Windows 7	Dell E5440	1	26.862	1 dual core	1.9GHz	8Gb

	Env	OS	hardware	min	seconds	processor(s)	speed	memory
5	BASH	**SuSE** Linux 42.2	Toshiba Portege R600	0	**0.478**	core2duo	1.4 GHz	4 Gb
9	**CygWin**	Windows 7	Dell E5440	1	26.862	1 dual core	1.9GHz	8Gb
4	**VM/Mac**	**Mint** Mate 17.3	MacBook Air 13" 2012	0	**0.227**	1	2GHz	8Gb
7	BASH	MacOSX 10.13.3	MacBook Air 13" 2012	0	1.941	1	2GHz	8Gb
1	BASH	**SuSE** Linux 42.1	Dell M4800	0	**0.202**	quad core	2.9 GHz	16 Gb
2	BASH	Ubuntu Linux 14.04	Dell T5810	0	**0.219**	4	3.7GHz	16 Gb
3	BASH	Centos7 Linux	Dell 7520	0	**0.263**	quad core	2.9 GHz	16 Gb
6	BASH	WINDOWS 10	Dell T5810	0	1.689	4	3.7GHz	16 Gb
8	**CygWin**	Windows 7	Dell T5810	0	5.110	4	3.7GHz	16 Gb

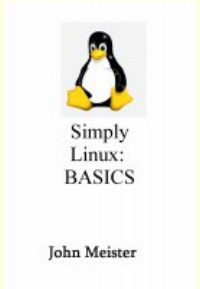
the BOTTOM LINE,
there are only a few command lines needed, e.g. with 4 reports:

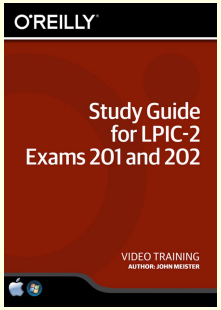
```
cat 01.txt | grep Mainten | grep - | awk '{print $4}' | sort | uniq | grep -v ^$ > 01.rpt
cat 02.txt | grep Mainten | grep - | awk '{print $4}' | sort | uniq | grep -v ^$ > 02.rpt
cat 03.txt | grep Mainten | grep - | awk '{print $4}' | sort | uniq | grep -v ^$ > 03.rpt
cat 04.txt | grep Mainten | grep - | awk '{print $4}' | sort | uniq | grep -v ^$ > 04.rpt
...
comm -12 01.rpt 02.rpt | comm -12 - 03.rpt | comm -12 - 04.rpt > common-errors.txt
cat common-errors.txt (find the ATA number)
grep -A 16 38-12044 ??.txt
        or:          grep -A 16 28-12044 [0-1][0-9].txt (many other options...)
```

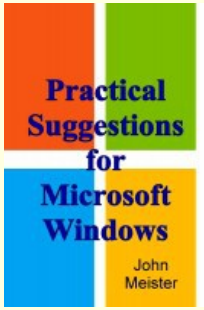
For another script example used for testing:
http://linuxmeister.net/Scripts/Engineering/wx-urls-sh-20jan2010.html

More LINUX info and resources:
http://linuxmeister.net/Intro-to-Linux/One-Hour-Linux-Sessions-2018.html

http://johnmeister.com/linux or http://LinuxMeister.net







Simply Linux: https://www.smashwords.com/books/view/705084
Using BASH on Win 10: https://www.smashwords.com/books/view/703463
Power Savings of Linux: https://www.smashwords.com/books/view/505731
Windows Suggestions: https://www.smashwords.com/books/view/508267
12 hour Video Course: **The Art of Linux System Administration**
published by O'Reilly Media Study Guide for the LPIC-2 Certification Exams

thank you....