Τμήμα Πληροφορικής
Πανεπιστήμιο Κύπρου

# ΕΠΛ232 – Προγραμματιστικές Τεχνικές και Εργαλεία

# Διάλεξη 22: Εργαλεία UNIX III

(Κεφάλαια 1-4 & 7, DAS-2ED)

**Δημήτρης Ζεϊναλιπούρ**

http://www.cs.ucy.ac.cy/courses/EPL232

# Περιεχόμενο Διάλεξης

- ***Έλεγχος Διεργασιών*** *(jobs, ps, fg, bg, kill, top),*
- ***Εύρεση Αρχείων και Προγραμμάτων*** *(which, whereis, find, exec, xargs),*
- ***Εντολές UNIX*** *(alias, cut, tr, tee, mail, comm, diff, crontab).*

# Έλεγχος Διεργασιών

- **Διεργασία:** Εάν πρόγραμμα υπό εκτέλεση
- Η εκτέλεση εντολών και προγραμμάτων προσθέτει **διεργασίες** στο σύστημα.
  - Π.χ., κάθε (υπό-)κέλυφος είναι μια διεργασία
- Ανά πάσα στιγμή στο UNIX εκτελούνται **πολλαπλές διεργασίες** (όχι κατ' ανάγκη ενεργές).
- Κάθε διεργασία αναγνωρίζεται από τον πυρήνα με το **Process IDentifier (PID)** αριθμό που ανατίθεται κατά την δημιουργία μιας διεργασίας.

# Έλεγχος Διεργασιών (Η εντολή **ps**)

- Μαθαίνοντας ποιες δι*εργασίες* τρέχουν
  - Εντολή *ps* (*process status*)
    - τυπώνει μια λίστα από *όλες* τις *διεργασίες* που βρίσκονται **υπό εκτέλεση** στο σύστημα.
    - δίνει μεταξύ άλλων το αναγνωριστικό **διεργασίας (*PID*)**, το **PPID parent PID που υποδηλώνει την διεργασία γονέα (που δημιούργησε την PID)**

Για να δούμε **ΚΑΘΕ (EVERY)** διεργασία με **standard σύνταξη**:

**ps –e ή ps –A**

Για να δούμε **ΚΑΘΕ (EVERY)** διεργασία με **BSD σύνταξη**:

**ps ax** (x: να περιλαμβάνει αυτές που δεν έχουν terminal)

** Οι επιλογές μεταξύ των συντάξεων μπορεί να αναμειγνύονται αλλά μπορεί να προκύψουν conflicts.*

# Έλεγχος Διεργασιών (Η εντολή **ps**)

*f: Full column format*

**Εκτέλεση από το κέλυφος: $ps -ef | head -15 (στο linux)**

Userid, processid, parentprocessid, cpu%, systemtime, command

| UID | PID | PPID | C | STIME | TTY | TIME | CMD |
|---|---|---|---|---|---|---|---|
| root | 1 | 0 | 0 | Feb15 | ? | 00:00:24 | init |
| root | 2 | 1 | 0 | Feb15 | ? | 00:00:00 | [keventd] |
| root | 3 | 1 | 0 | Feb15 | ? | 00:00:26 | [ksoftirqd_CPU0] |
| root | 4 | 1 | 0 | Feb15 | ? | 00:00:00 | [ksoftirqd_CPU1] |
| root | 5 | 1 | 0 | Feb15 | ? | 00:00:00 | [ksoftirqd_CPU2] |
| root | 6 | 1 | 0 | Feb15 | ? | 00:00:00 | [ksoftirqd_CPU3] |
| root | 7 | 1 | 0 | Feb15 | ? | 00:12:49 | [kswapd] |
| root | 8 | 1 | 4 | Feb15 | ? | 13:10:36 | [kscand] |
| root | 9 | 1 | 0 | Feb15 | ? | 00:00:00 | [bdflush] |
| root | 10 | 1 | 0 | Feb15 | ? | 00:00:32 | [kupdated] |
| | | | 0 | Feb15 | ? | 00:00:00 | [mdrecoveryd] |
| | | | 0 | Feb15 | ? | 00:00:00 | [scsi_eh_0] |
| | | | 0 | Feb15 | ? | 00:00:00 | [scsi_eh_1] |
| | | | 0 | Feb15 | ? | 00:02:59 | [kjournald] |

a) Η init γεννιέται από τον χρονοδρομολογητή του πυρήνα PID#0

b) Όλες οι διεργασίες «γεννιούνται» από την Init η οποια εχει PID#1

c) Οι διεργασίες αυτές εκτελούνται στο background και ονομάζονται daemon processes. Για αυτό τον λόγο δεν έχουν controlling terminal (stdin,out,err).

# Έλεγχος Διεργασιών (Η εντολή **ps**)

- Χρήσιμες επιλογές της *ps* εντολής
  - Για περισσότερα δείτε εγχειρίδιο (*man*)

| Flag | Meaning |
|---|---|
| a | All (BSD-syntax): Shows all processes associated with terminals attached to the system |
| u | User-Only (BSD-syntax): Produces output only for current user. |
| x | Terminal-Not-Necessary (BSD-syntax): Shows processes in the system including those with NO terminal. |
| -e \| -A | Every (Standard Syntax): Shows **every** processes on the system (same with “ax” option) |
| -f | Full (Standard Syntax): Gives full output format |

# Έλεγχος Διεργασιών (Η εντολή **ps**)

## See own processes with full-format output

```
bash-3.1$ ps fu
```

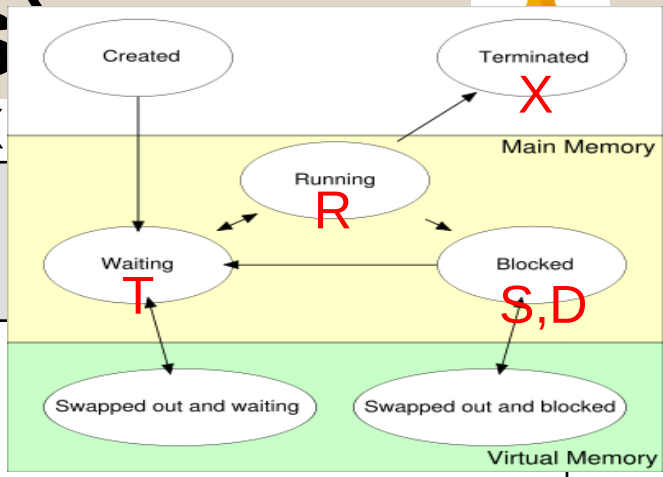
*Percent of CPU used* | *Percent of RAM used* | *Virtual memory used* | *Real memory used* | *Associated Terminal* | *Status (will study next)* | *Time process started* | *CPU Time used* | *Command Executed*

```
USER        PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
cspgcc1     808  0.0  0.1   6016  1572 pts/1    Ss   12:11   0:00 -ksh
cspgcc1     821  0.0  0.1   5580  1604 pts/1    S    12:11   0:00  \_ bash
cspgcc1    3386  0.0  0.0   5144   964 pts/1    R+   18:53   0:00      \_ ps fu
```

## See ALL processes on the machine

```
bash-3.1$ ps aux
USER        PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root          1  0.0  0.0   2008   680 ?        Ss   Jan30   0:01 init [5]
root          2  0.0  0.0      0     0 ?        S    Jan30   0:00 [migration/0]
.... . . . .
cs05ni1   32501  0.0  0.1   5092  1396 ?        Ssl  Feb02   0:00 dbus-daemon --
    f
cs05mp1     370  0.0  0.1   5088  1396 ?        Ssl  Feb02   0:00 dbus-daemon --
    f
ee06nn1     760  0.0  0.1  13284  1412 ?        Ssl  Feb02   0:00 dbus-daemon --
    f
cspgcc1     808  0.0  0.1   6016  1572 pts/1    Ss   12:11   0:00 -ksh
cspgcc1     821  0.0  0.1   5580  1604 pts/1    S    12:11   0:00 bash
cspgcc1    1307  0.0  0.0  13196   608 ?        Ssl  12:22   0:00 dbus-daemon --
    f
cspgcc1    3477  0.0  0.1   5180  1044 pts/1    R+   19:07   0:00 ps aux
```

# Έλεγχος Διεργασιών (Η εντολή **ps**)




## Process Status field with ps aux

| Status value | Meaning |
|---|---|
| R | **Running** or runnable (on run queue) |
| T | **Stopped**, either by a job control signal (CTRL-Z (20) or signal19) or because it is being traced. |
| S | **Interruptible sleep** (Blocked) (waiting for an event to complete) |
| D | **Uninterruptible sleep** (usually IO) |
| X | **Dead (Terminated):** Should never be seen |
| Z | **Zombie process** (the process has ended but hasn't returned the EXIT code to its parent, thus has not freed up its resources) **Θα επεξηγηθεί στην διάλεξη 13!** |

** Οι πιο πάνω κωδικοί μπορεί να συνοδεύονται από BSD-syntax κωδικούς, π.χ., "Ss" σημαίνει Blocked command + session leader*

# Έλεγχος Διεργασιών (Η εντολή **top και nice**)

- Μαθαίνοντας ποιες δι*εργασίες* τρέχουν
  - Εντολή *top*
    - Τυπώνει τις διεργασίες που χρησιμοποιούν το **μεγαλύτερο CPU χρόνο** (την πρώτη οθόνη μόνο η οποία **ενημερώνεται περιοδικά)**

```
bash-3.1$ top
top - 19:38:10 up 5 days,  5:10,  2 users,  load average: 0.04, 0.01, 0.00
Tasks: 115 total,   1 running, 113 sleeping,   1 stopped,   0 zombie
Cpu(s):  0.0% us,  0.2% sy,  0.0% ni, 99.8% id,  0.0% wa, 0.0% hi, 0.0% si,
Mem:   1025816k total,   926296k used,    99520k free,   145632k buffers
Swap:  1048568k total,        0k used,  1048568k free,   561744k cached

PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
  1 root      15   0  2008  680  588 S    0  0.1   0:01.31 init
  2 root      RT   0     0    0    0 S    0  0.0   0:00.00 migration/0
  3 root      34  19     0    0    0 S    0  0.0   0:00.00 ksoftirqd/0
  4 root      RT   0     0    0    0 S    0  0.0   0:00.00 watchdog/0
```

- 20 (highest priority)…+19 (lowest priority). Default: 0
nice -n 19 command (run command with certain nice level)
renice 19 -p 1024 (set nice level of running process)

RT: realtime priority process can never be pre-empted by timer interrupts and runs at a higher priority than any other thread in the system.

# Έλεγχος Διεργασιών (Η εντολή **watch**)

- Περιοδική προβολή μιας διοχέτευσης δείχνοντας την έξοδο στο output
  - Εντολή *watch (για οποιαδήποτε εντολή)*
    - Εξ' ορισμού κάθε 2 δευτερόλεπτα. ( **- n** )
    - Προβολή διαφορών ( **- d** )

```
bash-3.1$ watch -n 1 -d 'ps -e -o pid,uname,cmd,pmem,pcpu
   --sort=-pmem,-pcpu | head -5'

Every 1.0s: ps -e -o pid,uname,cmd,pmem,pcpu --sort=-pmem,-pcpu | head -5
   Mon Mar 30 17:34:55 2015

  PID USER     CMD                              %MEM %CPU
28458 kiacov02 /usr/lib64/firefox/firefox        7.7 11.5
21723 root     /usr/bin/perl /usr/lib64/x2       0.2  7.0
 1663 spolyk02 gnome-system-monitor --show       0.5  4.3
21731 root     /usr/bin/perl /usr/lib64/x2       0.2  4.0
```

# Τερματισμός Διεργασιών (Η εντολή **kill**)

- **Τερματίζοντας διεργασίες**
  - Εντολή *kill* <*options-number*> <*PID*>
    - στέλνει σήμα (μέσω πυρήνα) σε διεργασία που τρέχει
      - εξ' ορισμού, στέλνεται το σήμα τερματισμού SIGINT (TERM, Signal-2, Ctrl-C).
        - » Το σήμα-2 μπορεί να αγνοηθεί από την υπό εκτέλεση διεργασία.
      - Εάν θέλουμε οπωσδήποτε να τερματίσουμε μια διεργασία μπορούμε να στείλουμε το σήμα SIGKILL (KILL, Signal-9).
        - » Το σήμα-9 ΔΕΝ μπορεί να αγνοηθεί από μια διεργασία

# Έλεγχος Διεργασιών

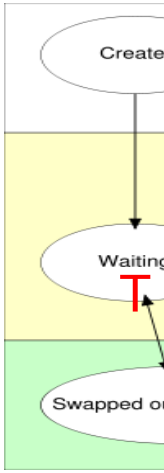
Αναμονή enter

```
bash-3.1$ ls -l ~ | grep test | sort | uniq | more &
[1] 3771
bash-3.1$

[1]+  Stopped                 ls -l ~ | grep test | sort | uniq | more
bash-3.1$
bash-3.1$ ps u
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
cspgcc1    808  0.0  0.1   6016  1572 pts/1    Ss   12:11   0:00 -ksh
cspgcc1    821  0.0  0.1   5580  1604 pts/1    S    12:11   0:00 bash
cspgcc1   3767  0.0  0.0   5320  1016 pts/1    T    20:14   0:00 ls -l /u/studen
cspgcc1   3768  0.0  0.0   4784   656 pts/1    T    20:14   0:00 grep test
cspgcc1   3769  0.0  0.0  30172   616 pts/1    T    20:14   0:00 sort
cspgcc1   3770  0.0  0.0   4520   428 pts/1    T    20:14   0:00 uniq
cspgcc1   3771  0.0  0.0   4536   476 pts/1    T    20:14   0:00 more
cspgcc1   3773  0.0  0.1   5180  1044 pts/1    R+   20:15   0:00 ps u
bash-3.1$
bash-3.1$ kill -9 3767
bash-3.1$ ps u
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
cspgcc1    808  0.0  0.1   6016  1572 pts/1    Ss   12:11   0:00 -ksh
cspgcc1    821  0.0  0.1   5580  1604 pts/1    S    12:11   0:00 bash
cspgcc1   3768  0.0  0.0   4784   656 pts/1    T    20:14   0:00 grep test
cspgcc1   3769  0.0  0.0  30172   616 pts/1    T    20:14   0:00 sort
cspgcc1   3770  0.0  0.0   4520   428 pts/1    T    20:14   0:00 uniq
cspgcc1   3771  0.0  0.0   4536   476 pts/1    T    20:14   0:00 more
cspgcc1   3775  0.0  0.1   5180  1048 pts/1    R+   20:15   0:00 ps u

[1]+  Stopped                 ls -l ~ | grep test | sort | uniq | more
```

Τοποθέτηση εντολής σε waiting T (λόγω more)




*Τι κάνει η εντολή kill -9 -1 και γιατί;*

# Τερματισμός Διεργασιών (Η εντολή **kill**)

- Η kill δεν είναι απλά εντολή τερματισμού μιας διεργασίας αλλά **εντολή αποστολής σήματος** σε μια διεργασία.
- **Παραδείγματα Σημάτων (θα δούμε κάποια από αυτά αργότερα στο μάθημα)**
  - SIGHUP (1) terminal line got hung-up (θυμηθείτε nohup)
  - **SIGINT (2) interrupt (Control^C)**
  - **SIGKILL (9) kill**
  - **SIGUSR1 (10) user-defined signal 1**
  - SIGSEGV (11) Segm. Viol. – invalid memory reference
  - **SIGUSR2 (12) user-defined signal 2**
  - SIGALRM (14) alarm clock
  - SIGTERM (15) software termination signal (like 2)
  - SIGCONT (18) continue after stop
  - SIGSTOP (19) stop **(cannot be caught or ignored).**
  - SIGTSTP (20) stop signal from keyboard (Control^Z)

# Διακοπή/Επαναφορά Διεργασιών (Η εντολή **fg**)

- Μπορούμε να σταματήσουμε (*προσωρινά*) μια *εργασία* (*εντολή γραμμής*) που τρέχει στο προσκήνιο επιλέγοντας το ***Ctrl-z* (σήμα-20)**
- Όταν μια **εργασία έχει σταματήσει**, μπορούμε να την επαναφέρουμε με:
  - Εντολή *fg (foreground)*
    - Ξεκινά (resume) ένα πρόγραμμα που είχε σταματήσει
    - Αναφέρεται στο πρόγραμμα που έχει έλεγχο του κελύφους

# Διακοπή/Επαναφορά Διεργασιών (Η εντολή **fg**)

- Παράδειγμα

**$ vi file.txt** *# Εδώ ανοίγει ο vi editor με το αρχείο file.txt*

~

~

Ctrl-Z # Εδώ το διακόπτουμε και επιστρέφουμε στο κέλυφος

**$ fg # Το επαναφέρουμε ξανά στο προσκήνιο**

~

~

- Μπορούμε να διακόψουμε πολλαπλές διεργασίες και να τις επαναφέρουμε μια-μια;
  - ΝΑΙ (δες επόμενη διαφάνεια)

# Διακοπή/Επαναφορά Διεργασιών (Η εντολή **fg**)

- Παράδειγμα

**$ vi file1.txt** *# Εδώ ανοίγει ο vi editor με το αρχείο file1.txt*

~

Ctrl-Z # Suspend

**$ bg**

**$ vi file2.txt** *# Εδώ ανοίγει ο vi editor με το αρχείο file2.txt*

~

Ctrl-Z # Suspend

**$ bg**

**$ fg** # επαναφέρει το αρχείο file2 (ανασύρεται από μια στοίβα!)

--- Εδώ κλείνουμε το αρχείο file2 με ESC-:-q!

**$ fg** # επαναφέρει το αρχείο file1 (ανασύρεται από μια στοίβα!)

--- Εδώ κλείνουμε το αρχείο file1 με ESC-:-q!

# Διακοπή/Επαναφορά Διεργασιών (Η εντολή **bg** και το **&**)

- Εντολή *bg (background)*
  - Τοποθετεί μια διεργασία στο background.

- **Χρήση του &:** Μια διαφορετική προσέγγιση είναι να **ξεκινήσουμε ένα πρόγραμμα** στο παρασκήνιο και να **αφήσουμε το UNIX σύστημα** να το διαχειριστεί.
  - Ένα πρόγραμμα (ή διοχέτευση) αυτόματα μπαίνει στο **παρασκήνιο**, τοποθετώντας ένα *&* στο τέλος της γραμμής εντολής.
  - Εάν το πρόγραμμα χρειάζεται κάποια είσοδο ή έξοδο, τότε σταματάει,
    - όπως οι εργασίες που έχουμε βάλει στο παρασκήνιο με την εντολή *bg* μετά που έχουν ήδη ξεκινήσει να τρέχουν.

# Διακοπή/Επαναφορά Διεργασιών (Η εντολή **bg** και το **&**)

```
bash-3.1$ ls -l ~ | grep test | sort | uniq | more &
[1] 2723
bash-3.1$

[1]+ Stopped ls -l ~ | grep test | sort |uniq | more
bash-3.1$
bash-3.1$ fg
ls -l ~ | grep test | sort | uniq | more
-rw-r--r-- 1 cspgcc1 cspg 0 Feb 1 00:22 test-cut.txt~
-rw-r--r-- 1 cspgcc1 cspg 0 Feb  1 00:30 test-tr~
-rw-r--r-- 1 cspgcc1 cspg 0 Jan 24 09:08 test1.txt
```

# Ακολουθιακή vs. Παράλληλη Εκτέλεση Εντολών

- Τρέχοντας πολλαπλές διεργασίες
  - **Ακολουθιακά**
    - Είναι δυνατό να υποδείξουμε στο UNIX να εκτελέσει μια σειρά εντολών χρησιμοποιώντας μια γραμμή εντολής κάνοντας χρήση του τελεστή «;»

    π.χ. `ls ; more *.txt ; cd`
  - **Παράλληλα**

    π.χ. `ls & cat file.txt & grep "pattern" file.txt`

# Διεργασίες (Process) vs. Εργασίες (Jobs)

- **Διεργασία (*Process*):**
  - Πώς το λειτουργικό σύστημα UNIX αναφέρεται σ' ένα πρόγραμμα που βρίσκεται υπό εκτέλεση.
  - Αναγνώριση από **PID**: τυχαίος αριθμός από τον πυρήνα του ΛΣ.
- **Εργασία (*Job*):**
  - Πώς το **κέλυφος** βλέπει τα προγράμματα που τρέχουν που **έχουν ξεκινήσει** από το **κέλυφος**
  - Αναγνώριση από **JobID**: αύξων αριθμός κελύφους
- ***Μια ολοκληρωμένη εντολή στη γραμμή εντολών του UNIX είναι μια εργασία.***

# Έλεγχος Εργασιών (Jobs) (Η εντολή **jobs**)

- Μαθαίνοντας ποιες *εργασίες* τρέχουν
  - Εντολή *jobs*
    - τυπώνει μια **λίστα από *όλες* τις *εργασίες*** που τρέχουν στο ***παρασκήνιο*** του **κελύφους**
    - δίνει τον **αριθμό ελέγχου εργασίας (*job ID*)** και την **γραμμή εντολής που τρέχει**.

# Έλεγχος Εργασιών (Jobs)

**JOB 1**

```
bash-3.1$ grep '^c.*h$' /usr/share/dict/words > output.txt &
// έστω ότι καθυστερεί ... και πληκτρολογούμε enter
[1]+  Stopped  grep '^c.*h$' /usr/share/dict/words >output.txt
```

\+ *υποδηλώνει την πιο πρόσφατη εργασία*

**JOB 2**

```
bash-3.1$ ls -l ~ | grep test | sort | uniq | more &
// αυτή η εργασία αναμένει enter για να προχωρήσει
[2] 3035
bash-3.1$

[2]+  Stopped    ls -l ~ | grep test | sort | uniq | more
bash-3.1$ jobs
[1]-  Stopped  grep '^c.*h$' /usr/share/dict/words | sort -r
   >output.txt
[2]+  Stopped    ls -l ~ | grep test | sort | uniq | more
```

# Έλεγχος Εργασιών (Jobs) (fg %, bg %, kill %)

- Στέλλοντας *εργασίες* στο **προσκήνιο**
  - Εντολή *fg %<job_ID>*
    - Η εντολή *fg* χωρίς παραμέτρους χειρίζεται την πιο πρόσφατη εργασία («σημαδεμένη» με ένα **+** στη λίστα των εργασιών)
- Στέλλοντας *εργασίες* στο **παρασκήνο**
  - Εντολή *bg %<job_ID>*
    - Η εντολή *bg* χωρίς παραμέτρους χειρίζεται την πιο πρόσφατη εργασία («σημαδεμένη» με ένα **+** στη λίστα των εργασιών)

# Έλεγχος Εργασιών (Jobs) (fg %, bg %, kill %)

```
bash-3.1$ fg %1
grep '^c.*h$' /usr/share/dict/words | sort -r
  >output.txt
```

Μετά

```
bash-3.1$ bg %1
[1]- grep '^c.*h$' /usr/share/dict/words |
  sort -r >output.txt &
bash-3.1$
[1]- Done  grep '^c.*h$' /usr/share/dict/words
  | sort -r >output.txt  # Εδώ ολοκληρώνει η
  εκτέλεση της εργασίας 1.

bash-3.1$ jobs
[2]+  Stopped                 ls -l ~ | grep
  test | sort | uniq | more
```

# Έλεγχος Εργασιών (Jobs) (fg %, bg %, kill %)

- Τερματίζοντας εργασίες
  - Εντολή *kill %* <*job_ID*>

**jobs -l**

```
[1]+  9856 Stopped (tty output)    ls --co
      9857                 | grep test
      9858                 | sort
      9859                 | uniq
      9860                 | more
```

**JOB 1**

```
bash-3.1$ ls -l ~ | grep test | sort | uniq | more &
[1] 3650
bash-3.1$ jobs
[1]+  Stopped                 ls -l ~ | grep test | sort | uniq | more
bash-3.1$
bash-3.1$ ps u
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
cspgcc1    808  0.0  0.1   6016  1572 pts/1    Ss   12:11   0:00 -ksh
cspgcc1    821  0.0  0.1   5580  1604 pts/1    S    12:11   0:00 bash
cspgcc1   3646  0.0  0.0   5152   836 pts/1    T    19:47   0:00 ls -l
                                                              /u/studen…
cspgcc1   3647  0.0  0.0   4780   652 pts/1    T    19:47   0:00 grep test
cspgcc1   3648  0.0  0.0  30176   624 pts/1    T    19:47   0:00 sort
cspgcc1   3649  0.0  0.0   4520   428 pts/1    T    19:47   0:00 uniq
cspgcc1   3650  0.0  0.0   4536   472 pts/1    T    19:47   0:00 more
cspgcc1   3652  0.0  0.1   5180  1048 pts/1    R+   19:47   0:00 ps u
bash-3.1$
bash-3.1$ kill %1
bash-3.1$ ps u
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
cspgcc1    808  0.0  0.1   6016  1572 pts/1    Ss   12:11   0:00 -ksh
cspgcc1    821  0.0  0.1   5580  1604 pts/1    S    12:11   0:00 bash
cspgcc1   3654  0.0  0.1   5180  1048 pts/1    R+   19:47   0:00 ps u
[1]+  Terminated              ls -l ~ | grep test | sort | uniq | more
bash-3.1$ jobs
bash-3.1$
```

# Διακρίβωση Ταυτότητας Εντολής (Η εντολή **which**)

- Εντολή *which* <*ProgramName*>
  - Αναφέρει **ποιο εκτελέσιμο** τρέχει όταν καλείς ένα πρόγραμμα (μπορεί να υπάρχουν πολλαπλά εκτελέσιμα εγκατεστημένα στο σύστημα)
  - Ψάχνει στο δικό ***PATH*** *του χρήστη*

```
bash-3.1$ which more
/bin/more
bash-3.1$ which ls
/bin/ls
bash-3.1$ which gran
which: no gran in
  (/usr/local/bin:/bin/:/usr/bin:/etc:/usr/sbin:/
  usr/ucb:/u/students/cs/pgrad/cspgcc1/bin:/usr/b
  in/X11:/sbin:.)
```

# Εύρεση Εντολών / Προγραμμάτων (Η εντολή **whereis**)

- Εντολή *whereis* <*ProgramName*>
  - Εντοπίζει **i)** το **εκτελέσιμο, ii)** τον **πηγαίο κώδικα** και **iii)** τη **σελίδα εγχειριδίου** για μια εντολή

```
bash-3.1$ whereis ls
ls: /bin/ls /usr/share/man/man1/ls.1.gz
  /usr/share/man/man1p/ls.1p.gz
```

# Γενική Αναζήτηση στο Filesystem (Η εντολή **find**)

- *find* <*path*> <*options*> <*expression*>
  - **Ανεύρεση αρχείων** με βάση διαφόρων στοιχείων (δέστε *man find*)
    - Όνομα
    - Χρόνος τροποποίησης
    - Δικαιώματα πρόσβασης
    - Μέγεθος αρχείου

# Γενική Αναζήτηση στο Filesystem (Η εντολή **find**)

– Παράδεγμα:

```
bash-3.1$ find ~ -name "test*.txt"
/home/faculty/cchrys/test/test1/test1.txt
/home/faculty/cchrys/test/test2/test2.txt
/home/faculty/cchrys/test/test.txt
```

– Εντολές Δράσης:

- *-print*
  - Απλά εκτυπώνει το όνομα του αρχείου που έχει βρεθεί
- *-exec <COMMAND> \;*
  - Εκτελεί την εντολή (*COMMAND*) για κάθε αρχείο που ταιριάζει

# Γενική Αναζήτηση στο Filesystem (Η εντολή **find**)

- Επιλογές
  - *-name*
    - Ταιριάζει το όνομα ενός αρχείου
  - *-iname*
    - Case-insensitive
  - *-type*
    - *d* → directory
    - *f* → regular file
    - *L* → symbolic link
  - *-atime N*
    - Τελευταία πρόσβαση στο αρχείο N μέρες πριν.
  - *-mtime N*
    - Τελευταία τροποποίηση στο αρχείο N μέρες πριν.

    - *N+ :* Περισσότερες από N μέρες πριν
    - *N- :* Λιγότερες από N μέρες πριν

# Γενική Αναζήτηση στο Filesystem (Η εντολή **find**)

– Παράδεγμα (με ίδιο αποτέλεσμα):

```
bash-3.1$ find ~ -name "test*.txt" -print
/home/faculty/cchrys/test/test1/test1.txt
/home/faculty/cchrys/test/test2/test2.txt
/home/faculty/cchrys/test/test.txt

bash-3.1$ find ~ -name "test*.txt" -exec echo {} \;
/home/faculty/cchrys/test/test1/test1.txt
/home/faculty/cchrys/test/test2/test2.txt
/home/faculty/cchrys/test/test.txt
```

– ***{}*** αντικαθίσταται από το όνομα του αρχείου που ανευρίσκεται από την find.

– Η επιλογή *-exec* επιτρέπει την εκτέλεση μιας εντολής σε όλα τα αρχεία που ταιριάζουν σε συγκεκριμένα κριτήρια.

# Ορίσματα Γραμμής Εισόδου (Η εντολή **xargs**)

– Εντολή *xargs* <*COMMAND*>
  - Παίρνει τιμές από το **ρεύμα εισόδου** και τις μετατρέπει σε **ορίσματα γραμμής εντολής** (*command line parameters*)
  - Χρησιμοποιείται όταν ένα πρόγραμμα εμφανίζει στην έξοδο μια λίστα, η οποία θα χρησιμοποιηθεί ως είσοδος σ' άλλο πρόγραμμα

Παράδειγμα:

```
find . –name "*.txt" | xargs grep 'Markus'
find . –name "*.txt" -exec grep 'Markus' {} \;
```

Τρέχει μια φορά για κάθε αποτέλεσμα της find.

περισσότερες φορές από το *xargs!*

# Άλλες Εντολές UNIX (Η εντολή **alias**)

- Εντολή *alias* <*NAME*>=“*COMMAND*”
  - «συντόμευση» εντολής με ένα όνομα

```
bash-3.1$ alias ll='ls -l'

bash-3.1$ ll test/
total 4
drwxr-xr-x 2 cspgcc1 cspg 22 Jan 24 10:42 test2
-rwxrwxr-x 1 cspgcc1 cspg  0 Jan 30 19:40 test.txt

bash-3.1$ alias
alias ll='ls -l‘

bash-3.1$ unalias ll
bash-3.1$ alias
```

# Άλλες Εντολές UNIX (Η εντολή **cut**)

- Εντολή *cut* (επιλογές ***-d, -f***)
  - Εμφανίζει επιλεγμένα **πεδία** των **γραμμών αρχείου στην έξοδο**
  - **Επιλογή *–d (delimiter)***
    - καθορισμός νέας οριοθέτησης πεδίων στις γραμμές, αντί του *tab (π.χ., ‘,’ ή άλλος χαρακτήρας)*
  - **Επιλογή *–f[1,][2,]...[#N] (field)***
    - Επιλογή συγκεκριμένων πεδίων
  - Αργότερα θα δούμε τον πιο ισχυρό επεξεργαστή **πεδίων** ροών **awk**

# Άλλες Εντολές UNIX (Η εντολή **cut**)

```
bash-3.1$ more test-cut.txt
Line number 1
Line number 2
Line number 3
Line number 4
bash-3.1$ cut -d' ' -f3 test-cut.txt
1
2
3
4
bash-3.1$ cut -d' ' --field=1,3 test-cut.txt
Line 1
Line 2
Line 3
Line 4
```

# Άλλες Εντολές UNIX (Η εντολή **tr**)

- Εντολή *tr* (επιλογές *-d, -s*) - Translate
  - **Μεταφράζει**, **συμπιέζει** ή/και **διαγράφει** χαρακτήρες της εισόδου. Μετατροπή του FROM string σε TO string.
  - Χρήση: ***tr [option] <FROM> <TO>***
  - **Επιλογή *–d (delete)***
    - Διαγράφει τους χαρακτήρες εισόδου που ορίζονται στο *FROM.*
  - **Επιλογή – *s (suppress repetition)***
    - Αντικαθιστά κάθε συνεχόμενο επαναλαμβανόμενο χαρακτήρα της εισόδου που ορίζεται στο *FROM*

# Άλλες Εντολές UNIX (Η εντολή **tr**)

```
bash-3.1$ more test-tr.txt
Thiss is the start of the novel i wrote.

bash-3.1$ tr 'i' 'I' < test-tr.txt
ThIss Is the start of the novel I wrote.

bash-3.1$ tr -d 'is' < test-tr.txt
Th  the tart of the novel  wrote

bash-3.1$ tr -s 's' < test-tr.txt
This is the start of the novel i wrote.
```

Αργότερα θα δούμε πιο δυναμικούς τρόπους αντικατάστασης σε ρεύματα με **sed.**

# Άλλες Εντολές UNIX (Η εντολή **tee**)

- Εντολή *tee*
  - Διαβάζει από το **ρεύμα εισόδου** και γράφει στο **ρεύμα εξόδου** ΚΑΙ σε **αρχείο** ταυτόχρονα
    - Τοποθετείς την *tee* εντολή οπουδήποτε σε μια *διοχέτευση* για αντιγράψει το ρεύμα εισόδου της *tee* εντολής σε αρχείο και στο ρεύμα εξόδου ή στο επόμενο βήμα της *διοχέτευσης.*

```
sort  somefile.txt | tee sorted_file.txt |  uniq -c
```

# Άλλες Εντολές UNIX (Η εντολή **mail**)

- Εντολή *mail* (επιλογές *-s*, *-cc*)
  - Αποστολή και παραλαβή ηλεκτρονικών μηνυμάτων

  - Επιλογή *-s*
    - Θέμα (Subject)
  - Επιλογή *-cc*
    - cc-address

```
mail –s “EPL371” –cc dzeina
  epl371@cs.ucy.ac.cy < elp371_syllabus.pdf
```

# Άλλες Εντολές UNIX (Η εντολή **comm**)

- Εντολή *comm*
  - Συγκρίνει δυο **ταξινομημένα** αρχεία γραμμή-γραμμή
  - Χωρίς επιλογές, παράγει ως έξοδο τρεις στήλες.
    - **Στήλη 1** περιέχει **γραμμές μοναδικές του αρχείου 1**
    - **Στήλη 2** περιέχει **γραμμές μοναδικές του αρχείου 2**
    - **Στήλη 3** περιέχει **γραμμές κοινές των δυο αρχείων**

**$cat file1**

a

b

c

**$cat file2**

a

b

b

c

```
$ comm file1 file2*
                a
                b
        b                (2nd occurrence)
                c
```

Επιλογή *-1*
Καταστέλλει (suppress) τις γραμμές μοναδικές για αρχείο 1

Επιλογή *-2*
Καταστέλλει (suppress) τις γραμμές μοναδικές για αρχείο 2

Επιλογή *-3*
Καταστέλλει (suppress) τις γραμμές που εμφανίζονται και στα δυο αρχεία

* Όνομα αρχείου «**-**» σημαίνει να διαβάσουμε από το ρεύμα εισόδου

# Άλλες Εντολές UNIX (Η εντολή **diff**)

- Εντολή *diff*
  - Βρίσκει διαφορές μεταξύ δυο αρχείων
  - Επιλογή *–i (case insensitive)*

```
$cat file1
x
y
z
$cat file2
x
y
y
z
$ diff file1 file2
2a3
> y
```

**Explanation:**
a:added, d:deleted, c:changed
<Original Line Number> a/d/c <Modified Line Number>
> Added Text
< Deleted Text

Examples: http://en.wikipedia.org/wiki/Diff

# Άλλες Εντολές UNIX (Η εντολή **diff**)

- Η λειτουργία της diff στηρίζεται στην επίλυση του προβλήματος **longest common subsequence**.
- Σε αυτό το πρόβλημα υπάρχουν 2 ακολουθίες:
  - **a b c d f g** h **j** q **z**
  - **a b c d** e **f g** i **j** k r x y **z**
- Το ζητούμενο είναι βρεθεί η μακρύτερη ακολουθία η όποια εμφανίζεται και στις 2 ακολουθίες (ενδέχεται να υπάρχουν πολλές LCSS ακολουθίες), π.χ.,
  - **a b c d f g j z**
- Η diff κατ' επέκταση δηλώνει με'-' ή '+' ποια αντικείμενα δεν εμφανίζονται ή εμφανίζονται στη LCSS ακολουθία
  - e h i q k r x y
  - \+ - + - + + + +

# Άλλες Εντολές UNIX (Η εντολή **crontab**)

- Εντολή *crontab*
  - Χρησιμοποιείται για χρονοδρομολόγηση (*scheduling*) εκτέλεσης εντολών, περιοδικά.
  - Οι εντολές μαζεύονται σε αρχείο γνωστό ως «**crontab**», το οποίο διαβάζεται και του οποίου οι εντολές τρέχουν στο παρασκήνιο από το ***cron daemon***
    - το οποίο τρέχει **σταθερά στο παρασκήνιο** και ελέγχει **κάθε λεπτό** να δει αν υπάρχουν εργασίες που πρέπει να εκτελεστούν.
  - Αυτές οι εντολές ονομάζονται ***cron jobs***.
  - Για να χρονοδρομολογήσεις τις εργασίες που θέλεις, πρέπει να τις αποστείλεις στο **cron daemon** χρησιμοποιώντας την εντολή **crontab**.
    - Η επόμενη διαφάνεια δείχνει πως

# Άλλες Εντολές UNIX (Η εντολή **crontab**)

- *crontab -e*
  - Edit your crontab file, or **create one** if it doesn't already exist.
- *crontab -l*
  - Display your contab file
- **Crontab σύνταξη :**
  - **Ορίζει πότε να εκτελείται η εργασία και ποια εργασία.**
  - Ένα **crontab** αρχείο έχει πέντε πεδία για ορισμό μέρας, ημερομηνίας και χρόνου, ακολουθούμενο από την εντολή που θα τρέχει στο καθορισμένο διάστημα

***  *  *  *  *  «command to be executed»**

```
 -   -   -   -   -
|   |   |   |   |
|   |   |   |   +----- day of week (0 - 6) (Sunday=0)
|   |   |   +------- month (1 - 12)
|   |   +--------- day of month (1 - 31)
|   +------------ hour (0 - 23)
+-------------- min (0 - 59)
```

# Άλλες Εντολές UNIX (Η εντολή **crontab**)

- Παράδειγμα Crontab
  - Μια γραμμή στο *crontab* αρχείο: Κάθε μέρα στις 6.32 μ.μ γράφει το περιεχόμενο του home καταλόγου σε αρχείο με τίτλο ***/home/someuser/tmp12342.out***, *όπου 12342 είναι το processID, που διεκπεραίωσε την εγγραφή.*

```
32  18   *  *   *   ls -al ~ > /home/someuser/tmp$$.out
```

*TIP: Εάν δεν δουλεύει μια cron εργασία μπορείτε να την ανακατευθύνεται το stdout, stderr σε αρχείο για να βρείτε το λάθος*