UOPR - Μάθημα 3ο

1. Διεργασίες
2. Χρονο-προγραμματισμός/ αυτοματοποίηση διεργασιών
3. Αναζήτηση αρχείων
4. Εντολές δικτύου και απομακρυσμένη σύνδεση SSH
5. Εντολές υλικού
6. Διαχείριση λογισμικού

---

1. Διεργασίες (Processes)

1α] Εισαγωγή

- Οι διεργασίες αποτελούν μέρος μιας εντολής ή προγράμματος που τρέχουν στο Η/Υ μας και απασχολούν πόρους του συστήματος (τον επεξεργαστή και την μνήμη RAM, τους δίσκους, την κάρτα δικτύου κτλ).
- Μια μόνο εντολή μπορεί να ξεκινήσει πολλές διεργασίες ταυτόχρονα. Ορισμένες διεργασίες είναι ανεξάρτητες μεταξύ τους, ενώ άλλες σχετίζονται. Μια αποτυχία σε μία διεργασία μπορεί, ή και να μη, επηρεάσει τις υπόλοιπες που εκτελούνται στο σύστημα.
- Το λειτουργικό σύστημα (και ειδικότερα ο πυρήνας) είναι υπεύθυνο για την κατανομή των πόρων του συστήματος σε κάθε διεργασία και για τη διασφάλιση της βέλτιστης λειτουργίας.

1β] Τύποι διεργασιών

Τύποι διεργασιών	Περιγραφή	Παράδειγμα
Διεργασίες χρηστών	Οι διεργασίες που ξεκινούν από τους χρήστες του συστήματος.	**bash**, **firefox**, **top**, **thunderbird**, **libreoffice**
Daemons/Services	Υπηρεσίες συστήματος που εκτελούνται συνεχώς. Πολλές από αυτές ξεκινούν κατά την εκκίνηση του συστήματος και στη συνέχεια περιμένουν ένα αίτημα από τον χρήστη ή το σύστημα που να υποδεικνύει ότι απαιτείται η υπηρεσία τους.	**httpd**, **sshd**, **libvirtd**, **cupsd**
Νήματα (threads)	Πρόκειται για εργασίες που εκτελούνται κάτω από μια κύρια διεργασία, μοιράζονται τη μνήμη	**dconf-service**, **gnome-**

Τύποι διεργασιών	Περιγραφή	Παράδειγμα
	και άλλους πόρους, αλλά προγραμματίζονται και εκτελούνται από το σύστημα ανεξάρτητα. Ένα μεμονωμένο νήμα μπορεί να τερματιστεί χωρίς να τερματιστεί ολόκληρη η διεργασία, και μια διεργασία μπορεί να δημιουργήσει νέα νήματα οποιαδήποτε στιγμή. Πολλά προγράμματα είναι multi-threaded.	**terminal-server**
Νήματα πυρήνα (Kernel Threads)	Εργασίες του πυρήνα που οι χρήστες έχουν ελάχιστο έλεγχο. Μπορεί να εκτελούν ενέργειες όπως η μεταφορά ενός νήματος από έναν επεξεργαστή σε έναν άλλο ή η διασφάλιση ότι οι λειτουργίες εισόδου/εξόδου προς τον δίσκο ολοκληρώνονται.	**kthreadd**, **migration**, **ksoftirqd**

Τύποι διεργασιών

1γ] Προβολή διεργασιών

Εντολή	Λειτουργία
`ps -u greeklug`	Προβολή των διεργασιών του χρήστη `greeklug`
`ps -e`	Προβολή όλων των διεργασιών
`pstree`	Προβολή όλων των διεργασιών σε μορφή "δέντρου"
`pstree -p`	Προβολή όλων των διεργασιών σε μορφή "δέντρου", με το PID
`top`	δυναμική προβολή των διεργασιών ανά 3"
`top -d = 30`	δυναμική προβολή των διεργασιών ανά 30"
`htop`	διαδραστικό πρόγραμμα προβολής διεργασιών

Εντολές προβολής διεργασιών

- Ανά πάσα στιγμή, εκτελούνται πολλαπλές διεργασίες. Το λειτουργικό σύστημα τις παρακολουθεί αναθέτοντας σε καθεμία έναν μοναδικό αριθμό αναγνώρισης διεργασίας (**PID**). Το PID χρησιμοποιείται για την παρακολούθηση της κατάστασης της διεργασίας, της χρήσης της CPU, της χρήσης μνήμης, της ακριβούς τοποθεσίας των πόρων στη μνήμη κα.
- Τα νέα PID αποδίδονται με αύξουσα σειρά καθώς γεννιούνται νέες διεργασίες. Το PID 1 αντιστοιχεί στη διεργασία **init** (systemd).
- Μια κρίσιμη λειτουργία του πυρήνα, που ονομάζεται **προγραμματιστής εργασιών (scheduler)**, μετακινεί συνεχώς διεργασίες εντός και εκτός του επεξεργαστή (CPU). Μοιράζει το χρόνο χρήσης του επεξεργαστή με βάση τη σχετική προτεραιότητα της κάθε διεργασίας.

Κατάσταση διεργασιών	Περιγραφή
κατάσταση εκτέλεσης (running)	Διεργασίες που εκτελούν αυτή τη στιγμή εντολές σε έναν επεξεργαστή, ή περιμένουν να τις παραχωρηθούν ένα μερίδιο χρόνου για να εκτελεστούν
κατάσταση ύπνου (sleep)	Διεργασίες που περιμένουν κάτι να συμβεί για να συνεχίσουν, πχ να πληκτρολογήσει κάτι ο χρήστης
κατάσταση zombie	Διεργασίες που έχουν ολοκληρώσει την εκτέλεσή τους, αλλά παραμένουν στη λίστα διεργασιών επειδή η γονική διεργασία (parent process) δεν έχει διαβάσει την έξοδό τους

Κατάσταση διεργασιών

Ανάγνωση του αποτελέσματος της εντολής `top`

```
top - 12:31:13 up 2 days, 22:35,  1 user,  load average: 0,48, 0,37, 0,44
Tasks: 285 total,   1 running, 282 sleeping,   0 stopped,   2 zombie
%Cpu(s):  1,8 us,  0,7 sy,  0,0 ni, 97,6 id,  0,0 wa,  0,0 hi,  0,0 si,
0,0 st
MiB Mem :  15875,1 total,   4411,5 free,   3444,4 used,   8581,2
buff/cache
MiB Swap:   1956,0 total,   1397,7 free,    558,2 used.  12430,6 avail Mem
```

1η γραμμή: τρέχουσα ώρα, uptime, αριθμός συνδεδεμένων χρηστών, load average
Το **load average** είναι η μέση χρήση του επεξεργαστή από διεργασίες σε ένα συγκεκριμένο χρονικό διάστημα. Η εκτέλεση των διεργασιών μοιράζεται σε όλα τα νήματα του επεξεργαστή. Για παράδειγμα, αν ένας επεξεργαστής έχει 6 πυρήνες (cores) και 12 νήματα (threads), το load average είναι η ποσοστιαία χρήση του επεξεργαστή διά 12. Αν ένας επεξεργαστής έχει 4 νήματα και το load average είναι 4, τότε αυτό σημαίνει ότι το σύστημα έχει πρόβλημα, καθώς ο επεξεργαστής χρησιμοποιείται κατά 100%!
Στο παραπάνω στιγμιότυπο, το load average σημαίνει ζητούσαν χρήση CPU το τελευταίο λεπτό 0,48 διεργασίες, 0,37 διεργασίες τα τελευταία 5' και 0,44 διεργασίες τα τελευταία 15'.

2η γραμμή: Συνολικός αριθμός των διεργασιών που είναι ενεργές.
3η γραμμή: Χρήση του επεξεργαστή από τον χρήστη (us), από τον πυρήνα (sy), ποσοστό διεργασιών με χαμηλή προτεραιότητα (ni), ποσοστό idle (id), ποσοστό διεργασιών που περιμένουν (wa).
4η γραμμή: Πληροφορίες και χρήση της μνήμης RAM.
5γραμμή: Πληροφορίες και χρήση της μνήμης Swap (χώρος στον δίσκο, που ενεργοποιείται όταν η μνήμη RAM δεν είναι αρκετή).

```
   PID  USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+
COMMAND
 213888 savvas    20   0   14576   5632   3456 R  15,4   0,0   0:00.04 top
   1855 savvas    20   0  898800  80404  43828 S   7,7   0,5   9:47.92
mintmenu        1 root      20   0   23112  13744   9136 S   0,0   0,1
0:13.45 systemd         2 root      20   0       0      0      0 S   0,0
0,0   0:00.13 kthreadd
```

- Ανά πάσα στιγμή, πολλές διεργασίες εκτελούνται (δηλαδή βρίσκονται στην ουρά εκτέλεσης) στο σύστημα. Ωστόσο, ένας επεξεργαστής μπορεί στην πραγματικότητα να εκτελεί μόνο μία εργασία τη φορά. Ορισμένες διεργασίες είναι πιο σημαντικές από άλλες, επομένως το Linux επιτρέπει τον καθορισμό και τη διαχείριση της προτεραιότητας των διεργασιών. Οι διεργασίες με υψηλότερη προτεραιότητα έχουν προνομιακή πρόσβαση στον επεξεργαστή.
- Η προτεραιότητα μιας διεργασίας μπορεί να καθοριστεί με τον ορισμό μιας **τιμής nice** (στήλη `NI` στο παραπάνω στιγμιότυπο), για τη συγκεκριμένη διεργασία.
- Όσο μικρότερη είναι η τιμή nice, τόσο υψηλότερη είναι η προτεραιότητα. Χαμηλές τιμές αποδίδονται σε σημαντικές διεργασίες, ενώ υψηλές τιμές σε διεργασίες που μπορούν να περιμένουν περισσότερο. Μια διεργασία με υψηλή τιμή nice απλώς επιτρέπει σε άλλες διεργασίες να εκτελεστούν πρώτες. Στο Linux, μια τιμή nice **-20** αντιπροσωπεύει τη μέγιστη προτεραιότητα και η τιμή **+19** τη χαμηλότερη.

1δ] Διαχείριση διεργασιών

- Μπορούμε να εκτελέσουμε μια διεργασία απευθείας από το τερματικό, πχ ανοίγουμε το πρόγραμμα geany
  `geany`
- Το πρόγραμμα τώρα είναι στο foreground και δεν μπορούμε να εκτελέσουμε/γράψουμε άλλη εντολή στο τερματικό.
- Πληκτρολογούμε `Ctrl + Z` για να βάλουμε την διεργασία σε παύση. Μέσα σε αγκύλη βλέπουμε το PID της (πχ PID = 2).
- Πληκτρολογώντας `bg` βάζουμε την διεργασία στο background και μπορούμε πάλι να εκτελέσουμε και άλλες εντολές στο τερματικό.
- Πληκτρολογώντας `fg` επαναφέρουμε την διεργασία στο foreground.
- Πληκτρολογώντας `sudo renice +10 2` αυξάνουμε το nice της διεργασίας 2 κατά 10, μειώνοντας την προτεραιότητά της.
- Πληκτρολογώντας `sudo kill -15 2` στέλνουμε αίτημα ώστε η διεργασία 2 να τερματιστεί με ευγενικό τρόπο, δίνοντάς της την ευκαιρία να καθαρίσει τους πόρους της και να αποθηκεύσει τυχόν δεδομένα αν χρειάζεται.
- Πληκτρολογώντας `sudo kill -9 2` τερματίζουμε βίαια την διεργασία 2.

2. Χρονο-προγραμματισμός/ αυτοματοποίηση διεργασιών

cron

- Το **cron** είναι ένα εργαλείο που βασίζεται στον χρόνο. Μπορεί να εκτελεί επαναλαμβανόμενες εργασίες σε συγκεκριμένες ώρες ή/και ημέρες. Το **cron** βασίζεται στο αρχείο ρυθμίσεων **crontab** (πίνακας cron), το οποίο περιέχει διάφορες εντολές που πρέπει να εκτελούνται σε προγραμματισμένες χρονικές στιγμές.
- Κάθε γραμμή σε ένα αρχείο **crontab** αντιστοιχεί σε μία εργασία (job) και αποτελείται από μια λεγόμενη **έκφραση CRON** (CRON expression), ακολουθούμενη από μια εντολή που θα εκτελεστεί.
- Για να ανοίξουμε το αρχείο crontab του χρήστη μας και να προσθέσουμε μια εργασία cron, πληκτρολογούμε: `crontab -e`
- Μια εργασία cron θα πρέπει να αποτελείται από τα ακόλουθα πεδία:

Πεδίο	Περιγραφή	Τιμή
MIN	Λεπτό	0 to 59
HOUR	Ώρα	0 to 23
DOM	Μέρα του μήνα	1-31
MON	Μήνας	1-12
DOW	Μέρα της εβδομάδας	0-6 (0 = Κυριακή)
CMD	Εντολή	Script προς εκτέλεση

Παράδειγμα1:

Πρώτα δημιουργούμε ένα απλό script σε ένα επεξεργαστή κειμένου, πχ nano editor:

```
#!/bin/bash

lshw >> /home/greeklug/scriptlshw.txt
```

Στο παραπάνω προβάλλουμε το υλικό του Η/Υ μας και κάθε φορά το προσθέτουμε (append) στο αρχείο lshw.txt
Αποθηκεύουμε το script ως scriptlshw.sh σε έναν κατάλογο, πχ /home/greeklug και του προσθέτουμε δικαιώματα εκτέλεσης, για να είναι εκτελέσιμο:
`chmod 764 scriptlshw.sh`

Μετά προγραμματίζουμε το παραπάνω script, ως μια εργασία cron. Ανοίγουμε το αρχείο crontab:
`crontab -e`

και προσθέτουμε στο τέλος, ως νέα σειρά:
`0 12 * 1 * /home/greeklug/scriptlshw.sh`

Στο παραπάνω, το scriptlshw.sh θα εκτελείται κάθε Ιανουάριο, στις 12 το μεσημέρι, ανεξάρτητα από την μέρα.

Παράδειγμα2:

Πρώτα δημιουργούμε το script στο nano:

```
#!/bin/bash

uptime >> /home/greeklug/scriptuptime.txt
```

Στο παραπάνω βλέπουμε πόση ώρα είναι ανοιχτό το σύστημά μας, καθώς και το load average τη συγκεκριμένη χρονική στιγμή. Στη συνέχεια, το προσθέτουμε (append) στο αρχείο uptime.txt .
Αποθηκεύουμε το script ως scriptuptime.sh σε έναν κατάλογο, πχ /home/greeklug και του προσθέτουμε δικαιώματα εκτέλεσης, για να είναι εκτελέσιμο:
`chmod 764 scriptuptime.sh`

Μετά προγραμματίζουμε το παραπάνω script, ως μια εργασία cron. Ανοίγουμε το αρχείο crontab:
`crontab -e`
και προσθέτουμε στο τέλος, ως νέα σειρά:
`0 3 1 * * /home/greeklug/scriptuptime.sh`

Στο παραπάνω, το scriptuptime.sh θα εκτελείται κάθε μήνα, στις 3 το πρωί, ανεξάρτητα από την μέρα.

Παράδειγμα3:

Πρώτα δημιουργούμε το script στο nano:

```
#!/bin/bash

tar cvzf /home/greeklug/Documents/downloads.tar.gz
/home/greeklug/Downloads/*
```

Στο παραπάνω συμπιέζουμε και δημιουργούμε ένα tar αρχείο, όλα τα αρχεία στις λήψεις μας (`~/Downloads`) μέσα σε έναν κατάλογο στο `~/Documents` με όνομα `downloads.tar.gz` .
Αποθηκεύουμε το script ως backup.sh σε έναν κατάλογο, πχ /home/greeklug και του προσθέτουμε δικαιώματα εκτέλεσης, για να είναι εκτελέσιμο:
`chmod 764 backup.sh`

Μετά προγραμματίζουμε το παραπάνω script, ως μια εργασία cron. Ανοίγουμε το αρχείο crontab:
`crontab -e`
και προσθέτουμε στο τέλος, ως νέα σειρά:
`0 4 * * 0 /home/greeklug/backup.sh`

Στο παραπάνω, το backup.sh θα εκτελείται κάθε Κυριακή, στις 4 το πρωί.

Για να δούμε όλες τις προγραμματισμένες εργασίες cron που έχουμε δημιουργήσει:
`crontab -l`

Για να διαγράψουμε μια προγραμματισμένη εργασία cron που έχουμε δημιουργήσει:
`crontab -r`

---

3. Αναζήτηση αρχείων

3α] locate

Για να χρησιμοποιήσουμε την locate, ανάλογα τη διανομή μας, κάνουμε εγκατάσταση τα πακέτα `mlocate` ή `plocate`.

Η εντολή **locate** χρησιμοποιεί μια βάση δεδομένων που δημιουργείται από ένα εργαλείο, το **updatedb**. Τα περισσότερα συστήματα Linux εκτελούν αυτόματα αυτή τη διαδικασία μία φορά την ημέρα μέσω cron ή υπηρεσίας συστήματος. Ωστόσο, μπορούμε να την ενημερώσουμε οποιαδήποτε στιγμή απλώς εκτελώντας το, προτού αναζητήσουμε κάτι με την locate:
`sudo updatedb`

Σύνταξη:
locate file/dir

Παράδειγμα
`locate file.txt`
`locate greeklug | grep "2025"`
`locate greeklug-*`

3β] find

Η εντολή **find** είναι ένα εξαιρετικά χρήσιμο και συχνά χρησιμοποιούμενο εργαλείο. Αναζητά αναδρομικά μέσα στο δέντρο του συστήματος αρχείων ξεκινώντας από έναν συγκεκριμένο κατάλογο και εντοπίζει αρχεία που πληρούν συγκεκριμένες συνθήκες. Ο προεπιλεγμένος κατάλογος αναζήτησης είναι πάντα ο τρέχων κατάλογος εργασίας.

Σύνταξη:
find /path/directory search_parametres (**Πρώτα ορίζω τον κατάλογο αναζήτησης και μετά**

τις παραμέτρους αναζήτησης)

Options	Λειτουργία
`-name`	Ορισμός συγκεκριμένου ονόματος αρχείου/καταλόγου
`-type f`	Ορισμός τύπου αρχείου αναζήτησης ως αρχείο (file)
`-type d`	Ορισμός τύπου αρχείου αναζήτησης ως κατάλογο (directory)
`-ctime [+/-]n`	Ορισμός χρόνου δημιουργίας αρχείου (created). Το `n` αντιπροσωπεύει μέρες
`-atime [+/-]n`	Ορισμός χρόνου πρόσβασης/ανάγνωσης αρχείου (accessed). Το `n` αντιπροσωπεύει μέρες
`-mtime [+/-]n`	Ορισμός χρόνου μετατροπής/εγγραφής αρχείου (modified). Το `n` αντιπροσωπεύει μέρες
`-cmin [+/-]m`	Ορισμός χρόνου δημιουργίας αρχείου (created). Το `m` αντιπροσωπεύει λεπτά
`-amin [+/-]m`	Ορισμός χρόνου πρόσβασης/ανάγνωσης αρχείου (accessed). Το `m` αντιπροσωπεύει λεπτά
`-mmin [+/-]m`	Ορισμός χρόνου δημιουργίας αρχείου (created). Το `m` αντιπροσωπεύει λεπτά
`-size [+/-]n`	Ορισμός μεγέθους αρχείου
`-exec command {} ";"`	Εκτελεί μια εντολή σε κάθε αρχείο που βρίσκει

Options για χρήση με την find

- Σχετικά με το option `-atime n`, `-mtime n` κτλ, μπορούμε να καθορίσουμε χρονικές τιμές (**n**), **+n** ή **-n**.
- `n` σημαίνει ακριβώς τόσες μέρες, `+n` σημαίνει πάνω από τόσες μέρες και `-n` σημαίνει κάτω από τόσες μέρες
- Σχετικά με το option `-size` μπορούμε να καθορίσουμε μονάδες όπως bytes (**c**), kilobytes (**k**), megabytes (**M**), gigabytes (**G**) κ.λπ. Όπως και με τις χρονικές τιμές που αναφέρθηκαν παραπάνω, τα μεγέθη αρχείων μπορούν να είναι ακριβείς αριθμοί (**n**), **+n** ή **-n**.

Παραδείγματα:

`find /home/greeklug -type f -name greeklug*.txt`
Αναζητεί μέσα στο `/home/greeklug` όλα τα αρχεία .txt που ξεκινούν με το όνομα greeklug και καταλήγουν σε .txt

`find /home -type d -name greeklug`
Αναζητεί μέσα στο `/home/` όλους τους καταλόγους με το όνομα greeklug

`find /home/greeklug/Videos -type f -size +1000M`
Αναζητεί μέσα στο `/home/greeklug/Videos` όλα τα αρχεία που έχουν μεγαλύτερο μέγεθος από 1000MB

`find /home/greeklug/.cache/mozilla/* -exec rm -r {} ";"`
Αναζητεί μέσα στο `/home/greeklug/.cache/mozilla/` όλα τα αρχεία και τα διαγράφει
(!) προσοχή με τις διαγραφές (!)

---

4. Εντολές δικτύου και απομακρυσμένη σύνδεση SSH

4α] Εντολές υλικού

Εντολή, option και argument	**Λειτουργία**
`ping greeklug.gr`	Διαγνωστικό εργαλείο που ελέγχει την προσβασιμότητα ενός server στο δίκτυο
`ip addr show`	Προβολή διευθύνσεων για κάθε κάρτα δικτύου του συστήματος
`dig greeklug.gr`	Εμφάνιση DNS εγγραφών/πληροφοριών ενός hostname/domain
`traceroute greeklug.gr`	Προβολή των διαδρομών των πακέτων δικτύου μέχρι τον server
`mtr greeklug.gr`	Συνδυασμός πληροφοριών των εντολών `ping` και `traceroute`
`netstat -i`	Εμφάνιση πίνακα με όλες τις διεπαφές δικτύου
`ip route`	Διαχείριση πινάκων δρομολόγησης
`nslookup`	Αναζήτηση πληροφοριών DNS
`wget`	Λήψη αρχείων/ιστοσελίδων

Χρήσιμες εντολές για προβολή πληροφοριών σχετικά με το δίκτυο

4β] Απομακρυσμένη σύνδεση SSH

Το **SSH** (Secure Shell) είναι ένα κρυπτογραφικό πρωτόκολλο δικτύου που επιτρέπει την ασφαλή λειτουργία υπηρεσιών δικτύου. Χρησιμοποιείται κυρίως για την ασφαλή απομακρυσμένη σύνδεση σε έναν άλλο Η/Υ-Server και η εκτέλεση εντολών σε αυτών.

Τα παρακάτω βασίζονται σε διανομές Debian/Ubuntu
Για να εκμεταλλευτούμε το SSH στο Linux, θα πρέπει να εγκαταστήσουμε το εργαλείο

openSSH στον τοπικό μας Η/Υ (αν δεν υπάρχει):
`sudo apt install openssh-client`

και αντίστοιχα να εγκαταστήσουμε το παρακάτω πακέτο στον απομακρυσμένο server:
`sudo apt install openssh-server`

Στη συνέχεια θα πρέπει να ξεκινήσουμε την υπηρεσία ssh στον server:
`sudo systemctl enable ssh`
Και να ελέγξουμε ότι όντως "τρέχει":
`sudo systemctl status ssh`

Είμαστε έτοιμοι να συνδεθούμε στον απομακρυσμένο server:
`ssh greeklug@170.140.200.128`
όπου ο `greeklug` είναι ο χρήστης στον οποίο συνδεόμαστε και το `170.140.200.128` είναι η ip του server.
Για την επαλήθευση πληκτρολογούμε τον κωδικό του χρήστη `greeklug`

Ο παραπάνω τρόπος επαλήθευσης, όμως, δεν είναι ιδιαίτερα ασφαλής, επομένως είναι καλό να τον αλλάξουμε.
Έτσι, δημιουργούμε κλειδιά κρυπτογράφησης (ένα δημόσιο και ένα ιδιωτικό), μέσω του αλγορίθμου Ed25519:
`ssh-keygen -t ed25519`
Έτσι δημιουργείται το id_ed25519 (ιδιωτικό κλειδί) και το id_ed25519.pub (δημόσιο κλειδί)

Από προκαθορισμένα τα 2 κλειδιά αποθηκεύονται στο ~/.ssh
Το δημόσιο κλειδί πρέπει να μεταφερθεί στον server, ενώ το **ιδιωτικό παραμένει στο τοπικό μηχάνημα** και δεν το δίνουμε πουθενά!!
Τα 2 αυτά κλειδιά μπορούν να χρησιμοποιηθούν και σε άλλες υπηρεσίες/server για ταυτοποίηση (github/gitlab, κτλ).

Για την χρήση του ιδιωτικού κλειδιού, αλλάζουμε τα δικαιώματά του, ώστε να έχουμε μόνο εμείς πρόσβαση:
`chmod 600 id_ed25519`

Τέλος, απενεργοποιούμε την επαλήθευση μέσω κωδικού στον server.
Ανοίγουμε το παρακάτω αρχείο:
`sudo nano /etc/ssh/sshd_config`
και αλλάζουμε την τιμή πρόσβασης με κωδικό ως εξής: `PasswordAuthentication no`
Θα χρειαστεί να εκτελέσουμε και μια επανεκκίνηση της υπηρεσίας ssh:
`sudo systemctl restart ssh`

Τώρα η επαλήθευση πραγματοποιείται μόνο μέσω των κλειδιών και μόνο αν κάποιος έχει στην κατοχή του το ιδιωτικό μας κλειδί μπορεί να έχει πρόσβαση στο απομακρυσμένο μηχάνημά μας.

Χρήση του Secure copy (scp)

Για να αντιγράψουμε κάτι από το τοπικό μηχάνημα στον server χρησιμοποιούμε το scp:
Παράδειγμα:
`scp -r /home/savvas/ greeklug@172.159.200.143:/home/greeklug`
Στο παραπάνω αντιγράφουμε όλο το home του χρήστη `savvas` στο τοπικό Η/Υ, προς το home του χρήστη `greeklug` μέσα στον server.

---

5. εντολές υλικού

Εντολή, option και argument	**Λειτουργία**
`cat /proc/cpuinfo`	Αναλυτικές πληροφορίες για τον επεξεργαστή
`cat /proc/meminfo`	Αναλυτικές πληροφορίες για την RAM
`lshw`	Αναλυτικές πληροφορίες για όλο το υλικό του Η/Υ
`lspci`	Προβολή πληροφοριών για όλες τις συσκευές PCI
`lsusb`	Προβολή πληροφοριών για όλες τις συσκευές USB
`uname -a`	Προβολή όλων (`-a`) των πληροφοριών για τον πυρήνα
`sudo fdisk -l`	Προβολή πινάκων διαμερισμάτων, συνδεδεμένων συσκευών
`du -ha` `du -ha /home`	Εκτιμώμενη χρήση σε χώρου αρχείων και καταλόγων σε human readable format (`-h`)
`df -Th`	Πληροφορίες για τον αποθηκευτικό χώρο κάθε συνδεδεμένης κατάτμησης/συσκευής, μαζί με το filesystem του (`-T`) σε human readable format (`-h`)
`uptime -p`	Πόσο χρονικό διάστημα είναι ανοιχτό το σύστημα
`uptime -s`	Πότε άνοιξε το σύστημα
`free -h`	Σύντομη προβολή πληροφοριών για την RAM σε human readable format (`-h`)
`sensors`	Αναλυτικές πληροφορίες για αισθητήρες του συστήματος (θερμοκρασίες κτλ)
`inxi -Fxxxzr`	Περίληψη μερικών από τα παραπάνω

Χρήσιμες εντολές για προβολή πληροφοριών σχετικά με το υλικό

---

6. Διαχείριση Λογισμικού

Κάθε διανομή Linux χρησιμοποιεί ένα σύστημα πακέτων για την διαχείριση του λογισμικού. Τα ποιο γνωστά είναι το DEB (διανομές της οικογένειας Debian/Ubuntu) και το RPM

(διανομές οικογένειας RedHat/OpenSUSE).

- Τα πακέτα DEB έχουν την κατάληξη .deb και διαχειρίζονται μέσω εργαλείων όπως το dpkg, apt και apt-get.
- Τα πακέτα RPM έχουν την κατάληξη .rpm και διαχειρίζονται μέσω εργαλείων όπως το rpm και το dnf ή yum.

Κάθε διανομή διατηρεί επίσημα αποθετήρια των προγραμμάτων της, ώστε να μπορεί κάποιος να τα κάνει λήψη από τους επίσημους server και mirrors.
Διαφορετικές διανομές χρησιμοποιούν διαφορετικά συστήματα πακετοποίησης και, ως γενικός κανόνας, ένα πακέτο που προορίζεται για μία διανομή δεν είναι άμεσα συμβατό με κάποια άλλη διανομή. Οι περισσότερες διανομές ανήκουν σε μία από τις δύο παραπάνω βασικές κατηγορίες τεχνολογιών πακετοποίησης.
Τα τελευταία χρόνια, έχουν κάνει την εμφάνισή τους νέα συστήματα πακέτων που προσπαθούν να συμπεριλάβουν όλα τα στοιχεία του λογισμικού ώστε να μπορούν να εκτελεστούν σε όλες τις διανομές Linux.

Συνεπώς, σήμερα υπάρχουν πολλές πηγές λογισμικών σε Linux. Αυτές μπορεί να είναι:

- Επίσημα αποθετήρια διανομών (ενεργά από προκαθορισμένα)
- Επίσημα πρόσθετα/δοκιμαστικά αποθετήρια (συνήθως απαιτείται να τα ενεργοποιήσουμε για πρόσθετα ή νεότερα δοκιμαστικά πακέτα)
- Τρίτα αποθετήρια, πχ τα PPAs (Personal Package Archive) σχεδιασμένα για Ubuntu based διανομές
- AppImage (εκτελέσιμα αρχεία, χωρίς εγκατάσταση)
- Flatpaks (αυτοδύναμα πακέτα ανεξάρτητα απ' τις διανομές, λειτουργούν ως απομονωμένα από το σύστημα)
- Snaps (αυτοδύναμα πακέτα για Ubuntu based διανομές, περιέχουν όλες τις εξαρτήσεις όπως βιβλιοθήκες σε αντίθεση με τα DEB που μπορεί να χρειάζονται πρόσθετα πακέτα)

Διαχειριστές πακέτων: Δύο επίπεδα

Υπάρχουν 2 επίπεδα διαχείρισης λογισμικού: ένα εργαλείο χαμηλού επιπέδου (όπως το **dpkg** ή το **rpm**), που φροντίζει για τις λεπτομέρειες της αποσυμπίεσης των πακέτων, της εκτέλεσης σεναρίων και της σωστής εγκατάστασης του λογισμικού, ενώ ένα εργαλείο υψηλού επιπέδου (όπως το **apt**, το **dnf** ή το **zypper**) εργάζεται με **ομάδες πακέτων**, κατεβάζει πακέτα από τον πάροχο και **επιλύει τις εξαρτήσεις**.

Τις περισσότερες φορές, οι χρήστες χρειάζεται να χρησιμοποιούν **μόνο το εργαλείο υψηλού επιπέδου**, το οποίο φροντίζει να καλεί το εργαλείο χαμηλού επιπέδου όταν χρειάζεται. Η επίλυση εξαρτήσεων είναι ιδιαίτερα σημαντική δυνατότητα του εργαλείου υψηλού επιπέδου, καθώς αναλαμβάνει να βρει και να εγκαταστήσει αυτόματα κάθε απαραίτητο εξαρτώμενο πακέτο.

Linux Package Manager	Debian	SUSE	Red Hat
High Level Package Manager	apt	zypper	dnf
Low Level Package Manager	dpkg	rpm	rpm

2 επίπεδα διαχείρισης λογισμικού

Λειτουργία	Debian based	Red Hat based	Open Suse based	Arch based
Αναζήτηση πακέτου	`apt search nano`	`dnf search nano`	`zypper search nano`	`pacman -Ss nano`
Εγκατάσταση πακέτου (με εξαρτήσεις)	`apt install geany`	`dnf install nano`	`zypper install nano`	`pacman -S nano`
Αφαίρεση πακέτου (με εξαρτήσεις)	`apt remove nano`	`dnf remove nano`	`zypper remove nano`	`pacman -R nano`
Λήψη ενημερώσεων όλου του συστήματος	`apt update`	`dnf check-update`	`zypper refresh`	`pacman -Sy`
Αναβάθμιση όλου του συστήματος	`apt upgrade`	`dnf upgrade`	`zypper update`	`pacman -Syu`

Εντολές διαχείρισης λογισμικού στις πιο γνωστές οικογένειες διανομών. Ως παράδειγμα χρησιμοποιείται το πρόγραμμα `nano`. Οι παραπάνω εντολές απαιτούν δικαιώματα διαχειριστή, επομένως χρειάζεται το `sudo` πριν την καθεμιά.
