nedeljainformatike v2.0

Profesionalni razvoj softvera: Unix alati i `git`

Miloš Stanojević

Matematička gimnazija
Nedelja informatike v2.0

15. decembar 2015.

Unix shell

- Korisnički program koji Unix automatski pokreće nakon uspešnog logovanja;
- Omogućava korisniku da interaktivno pokreće i zaustavlja druge programe;
- Kontroliše pristup drugih programa terminalu;
- Omogućava automatizaciju (shell skripte);
- Omogućava konstrukcije kao programski jezici (npr. varijable, izraze sa stringovima, uslovno grananje, petlje, konkurentnost);
- Olakšava odabir fajlova preko tastature (regularni izrazi, file name completion);
- Olakšava unos komandnih argumenata preko modifikacionih funkcija i funkcija istorije;
- Najpoznatiji shell (“sh”) razvio je 1975. Stephen Bourne, a malo modernija GNU verzija je “bash” (“Bourne-Again SHell”).

File deskriptori

- File deskriptor (u daljem tekstu `fd`) je celobrojna vrednost[1] koju Unix asocira sa otvorenim fajlom (bilo kog tipa);
- Kada otvorimo neki fajl, kernel pripremi odgovarajuće strukture podataka, a nama prosledi file deskriptor koji se odonsi na taj, sada otvoreni fajl;
- Dakle, kako bismo pristupili nekom otvorenom fajlu, dovoljno je da kao jedan od argumenata funcijama za pristup fajlovima prosledimo odgovarajući `fd`.

[1] Postoje dve bitne tabele koje sistem čuva bez našeg znanja: per-process `fd` tabela i system-wide `fd` tabela. U system-wide `fd` tabeli se čuvaju veze ka otvorenim fajlovima svih procesa, a u per-process `fd` tabeli se čuvaju indeksi system-wide tabele koji odgovaraju otvorenim fajlovima nekog procesa. Konačno, sami file dekriptori su upravo indeksi per-process `fd` tabele nekog procesa.

File deskriptori, cont'd

Unix procesi pristupaju fajlovima na sledeći način:

- U okviru poziva funkcija `open()` ili `create()`, proslede kernelu putanju fajla, a kao povratnu vrednost dobiju celobrojni `fd`;
- U okviru sistemskih poziva `read()`, `write()` ili `seek()`, proslede ranije otvoreni `fd`, zajedno sa podacima;
- Konačno, pozivaju `close()` sa `fd`om kao parametrom, kako bi oslobodili strukture podataka vezane za odgovarajući otvoreni fajl.

Tri osnovna streama

Streamovi su tip komunikacionih kanala programa, sa okruženjem u kojem se on izvršava.

Po konvenciji, shell otvara tri osnovna streama za svaki proces, kojima odgovaraju `fd`ovi:

- 0 = standardni unos (`stdin`, za unos podataka koji će biti obrađeni)
- 1 = standardni izlaz (`stdout`, za ispis rezultata obrade)
- 2 = standardni error (`stderr`, za error poruke)

Pajpovi - nadovezivanje streamova

Postoji više načina da povežemo `stdin`, `stdout` i `stderr` pri pokretanju programa.

Osnovni način je naravno nadovezivanje standardnih streamova na terminal:

```
$ command
```

Ali lepota Unixa je u nečemu što zovemo **pajpovanje**, odnosno nadovezivanje `stdouta` jednog procesa direktno na `stdin` drugog procesa (upotrebom "cevi" odnosno **pajpa**). Na primer:

```
$ command1 | command2 | command3
```

povezuje `stdout` komande1 na `stdin` komande2 i `stdout` komande2 na `stdin` komande3, a preostale standardne streamove povezuje na terminal.

Preusmeravanje streamova - osnove

Moguće je takođe, poslati `stdout` u fajl:

```
$ command >filename
```

Ili, nadovezati `stdout` na sadržaj fajla:

```
$ command >>filename
```

Možemo učitavati vrednosti iz fajla u `stdin`:

```
$ command <filename
```

Takođe je moguće napraviti i komplikovanije konstrukcije:

```
$ command >filename 2>&1
```

Ova komanda šalje `stdout` i `stderr` u isti fajl, tako što prvo preusmeri `stdout` u `filename`, pa onda preusmeri `stderr` (`fd 2`) tamo gde smo poslali `stdout` (`&1` = vrednost na koju pokazuje `fd 1`).

Preusmeravanje streamova - naprednije

Možemo otvoriti i druge fdove za input, output ili oba:

```
$ command 0<in 1>out 2>>log 3<auxin 4>auxout 5<>data
```

"Here Document" nam omogućavaju da unosimo podatke u shell skripte direktno, tako što sa komandne linije unosimo tekst, a shell to preusmerava na stdin programa. Primer:

```
$ tr <<THEEND A-MN-Za-mn-z N-ZA-Mn-za-m
> Vs lbh zhfg cbfg n ehqr wbxr ba HFRARG, ebgngr gur
> nycunorg ol 13 punenpgref naq nqq n jneavat.
> THEEND
```

Bitno je da direktno iza << navedemo string koji će označavati kraj teksta.

Argumenti na komandnoj liniji

Svaki proces dobija od programa koji ga poziva sledeće parametre:

- `argc` = niz stringova koji sadrži sve argumente koji se prosleđuju procesu
- `**argv` = ceo broj prosleđenih stringova u argv
- `**environ` = opcioni niz strignova koji predstavljaju environment varijable (više o ovome kasnije)

Argumenti na komandnoj liniji - argv

Reči unesene nakon imena komande shell prvo prepocesira u par koraka, a nakon toga ih stavlja u argv parametre i prosleđuje samom procesu

```
$ cp 'Nedelja Informaike.pdf' nedelja-informatike-v40.pdf
$ mv *.bak useless-old-files/
```

Među argumentima možemo razlikovati **komandne opcije** od **parametara**.

Argumenti na komandnoj liniji - argv cont'd

Komandne opcije obavezno počinju sa:

- '-' (npr. "-h"), što znači da je opcija data u obliku jednog karaktera. Kada prosleđujemo opcije u ovom formatu, smemo napisati više karaktera za redom iza jednog '-' npr:

```
$ ls -l -a -t
$ ls -lat
```

- "--" (npr. "--help"), što znači da je opcija data u obliku reči. U ovom formatu ne smemo slepljivati opcije!

```
$ gcc --version
$ curl --head http://www.mg.edu.rs/
```

Ostali argumenti na komandnoj liniji uglavnom predstavljaju **parametre**, odnosno imena fajlova, URLove, itd.

Shell preprocesiranje na komandnoj liniji

Postoji nekoliko interpunkcijskih znakova na komandnoj liniji koji su deo kontrolne sintakse:

`| & ; ( ) < >`

ili mogu pokrenuti specialne zamene pre nego što se argv prosledi programu:

- pathname expansion / filename matching: `* ? []`
- quote removal: `\' "`
- brace expansion: `{,}`
- tilde expansion: `~`
- parameter/command expansion: `$ ‘`

Pathname expansion

Argumenti na komandnoj liniji koji sadrže `*`, `?` ili `[...]` tumače se kao regularni izrazi i biće zamenjeni sa listom svih odgovarajućih imena fajlova. Znake tumačimo kao:

- `?` je proizvoljni (jedan) karakter
- `*` je niz od nula ili više proizvoljnih karaktera
- `[...]` predstavlja karakter iz datog skupa. Koristimo "-" kako bismo precizirali opseg karaktera i "^" kako bisto taj skup negirali po potrebi.

Ništa od gorenavedenog neće biti upareno sa tačkom na početku imena fajla, jer je to konvencija za nazivanje skrivenih fajlova! Primeri:

```
*.bak [A-Za-z]*.??? [^A-Z] .??* files/*/*.o
```

quote removal

Postoje tri mehanizma za unošenje specijalnih karaktera na komandnoj liniji bez pokretanja njima odgovarajućih shell substitucija:

- ’ . . . ’ uklanja značenje svih specijalnih znakova
- " . . . " uklanja značenje svih specijalnih znakova, sem $ ` \ ‘
- \ klanja značenje svih specijalnih znakova za karakter odmah iza sebe

Ostale specijalne zamene

- brace expansion: Omogućava lak unos reči sa ponavljajućim podstringovima
- tilde expansion: Omogućava lak način za unos putanje `home` direktorijuma
- parameter expansion: Zamenjuje se sa vrednošću shell varijable ili `stdout`om neke komande

Uvedimo prvu alatku!

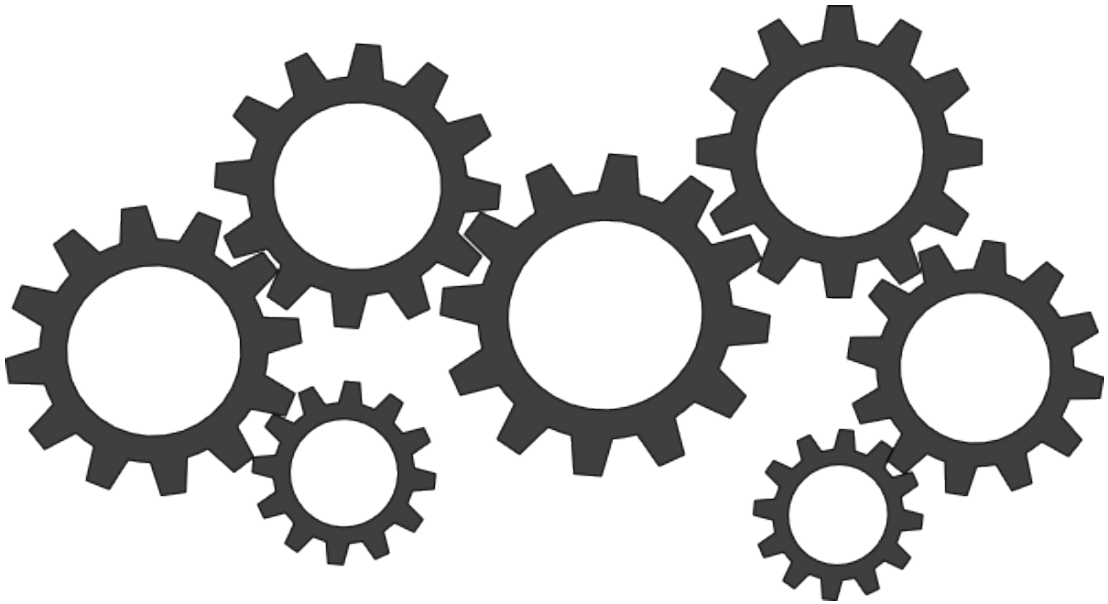
Pre nego što uradimo prvi demo, uvedimo najprostiji od svih Unix alata: `echo`

- Jedino što `echo` radi je vraća sve svoje parametre na standardni izlaz
- Na primer:

```
$ echo 'Zdravo svima!'
Zdravo svima!
```

Sa ovim znanjem, možemo preći u komandnu liniju i dati primere preprocesiranja na komandnoj liniji …

Demo preprocesiranja na komandnoj liniji



Osnove

Služe i kao promenljive (tipa string) u shell programiranju, ali i kao promenljive okruženja, u svrhu komunikacije sa drugim programima.

Ovako postavljamo *promenljivu* na *vrednost*:

```
$ variable=value
```

Obavezno bez razmaka pre i posle znaka jednakosti!

Promenljivu možemo učiniti vidljivom programima koje pozivamo:

```
$ export variable
$ export variable=value
```

Ili lokalno proslediti pozvanom programu:

```
$ variable1=value variable2=value command
```

Standardne Shell varijable

- `$HOME` – Naš “home” direktorijum.
- `$USER`/`$LOGNAME` – Korisničko ime.
- `$PATH` – Lista dvotačkom odvojenih direktorijuma u kojima shell traži komande (npr. “/bin:/usr/bin:/usr/X11R6/bin”)
- `$LD_LIBRARY_PATH` – Lista dvotačkom odvojenih direktorijuma gde loader traži zajedničke biblioteke.
- `$LANG`, `$LC_*` – Naš “locale”, odnosno sistemski konfiguracioni fajl sa informacijam o setu karaktera i konvencija u vezi sa jezikom i zemljom (npr. “en_GB.UTF-8 ”). `$LC_*` postavlja locale za samo jednu od navedenih kategorija (npr. `$LC_CTYPE` za set karaktera), dok `$LANG` postavlja vrednost svega.

man

- Verovatno najkorisnija alatka za početak
- Otvara detaljan "priručnik" funkcije za koju je pozvan:

```
$ man sudo
```

  Ova funkcija pronalazi uputstvo za potrebu komande `sudo` na `$MANPATH`, nakon čega otvara dve dodatne alatke (`nroff` text formatter i `more` text-file viewer) pomoću kojih to uputstvo predstavlja korisniku.
- Možemo odabrati sekcije priručnika `man`: user commands (1) – default, system calls (2), library functions (3), devices (4), file formats (5). Na primer

```
$ man 5 sudo
```

  otvara stranu priručnika koja se tiče formata fajlova koje funkcija prima odnosno vraća.

sudo

- Skraćeno od Super User DO
- Daje mogućnosti root usera nekoj funkciji:

PAŽNJA! VEOMA OPASNO!

```
$ sudo rm -rf /
```

Šta radi ova komanda?

- Moramo biti **veoma pažljivi** kada koristimo sudo!
- Najbolje ne koristiti stvari kao:

```
$ sudo su
```

već pažljivo davati zasebnim komandama root privilegije!

cd

- Skraćeno od Change Directory
- Ukratko, koristi se kako bismo promenili trenutni radni direktorijum
- Neke korisne opcije:
  - `..` – vraća se na roditeljski direktorijum.
- **Korisno:** TAB dugme je veoma korisno na Unix-based platforma, jer pokreće **auto completion**! Tačnije, nastavlja tekst koji trenutno kucamo do prve smislene vrednosti, a kada ga pritisnemo svaki narednu put, dopuniće naš tekst do sledeće mogućnosti, i tako u krug.

`ls` i `pwd`

- U izvornoj formi, `ls` izlistava sve direktorijume i fajlove koji se nalaze u trenutnom radnom direktorijumu
- Neke korisne opcije za `ls`:
  - `-lh` – daje detalje o veličini (`-h` znači human readable), vremenu modifikacije, imenu vlasnika itd.
  - `-a` – izlistava sve skrivene fajlove (one koji počinju sa tačkom)
  - `-lS` – izlistava fajlove sortirane u opadajućem redosledu po veličini
- `pwd` prosto ispisuje trenutni radi direktorijum na `stdout`

touch

- `touch` koristimo kako bismo napravili fajl, modifikovali ga ili mu promenili vreme pristupa
- Neke korisne opcije:
  - `-c` – ukoliko fajl ne postoji napraviće ga
  - `-a` – menjamo samo vreme pristupa
  - `-m` – menja samo vreme modifikacije
  - `-d` – menja i vreme pristupa i vreme modifikacije
  - `-t` – pravi fajl sa koristeći specificirano vreme (u formatu YYYYDDHHMM.SS)
  - `-r` – koristi vremena pristupa i modifikacije drugog fajla
- Bez ikakvih navedenih opcija, `touch` pravi prazan fajl.

mv, cp i rm

- `mv` služi za preimenovanje postojećih fajlova:

  ```
  $ mv starifajl novifajl
  ```

  će preimenovati postojeći fajl `starifajl` u `novifajl`.
- `cp` služi za kopiranje postojećih fajlova:

  ```
  $ cp starifajl novifajl
  ```

  će prekopirati postojeći fajl `starifajl` u `novifajl`.
- `rm` služi za brisanje postojećih fajlova:

  ```
  $ rm starifajl
  ```

  će obrisati postojeći fajl `starifajl`. Korisna opcija: `-i` omogućava da `rm` pita pre nego što obriše svaki od zadatih fajlova.

mkdir i rmdir

- `mkdir` koristimo kako bismo napravili direktorijum
- `rmdir` koristimo kako bismo obrisali direktorijum

cat

- Skraćeno od conCATenate
- Ima široku upotrebu koja uključuje spajanje tekstualnih fajlova, pravljenje novih ali i njihovu modifikaciju
- U osnovnom obliku:

```
$ cat nekifajl
```

ispisuje sadržinu datog fajla na standardni output.

- Možemo pisati i direktno na kraj nekog fajla:

```
$ cat >>nekifajl
```

Sve što otkucamo iza gorenavedene komande biće dopisano u `nekifajl` nakon što pritisnemo Ctrl+D.

- Takođe, možemo spojiti sadržine fajlova `prvifajl` i `drugifajl` u `trecifajl`:

```
$ cat prvifajl drugifajl >trecifajl
```

less

- Koristi se kako bismo pregledali fajlove, umesto da ih otvaramo
- Omogućava kretanje po fajlu, i ne mora da učita ceo fajl odjednom, tako da se brzo otvara
- Da bismo otvorili neki fajl:

```
$ less nekifajl
```

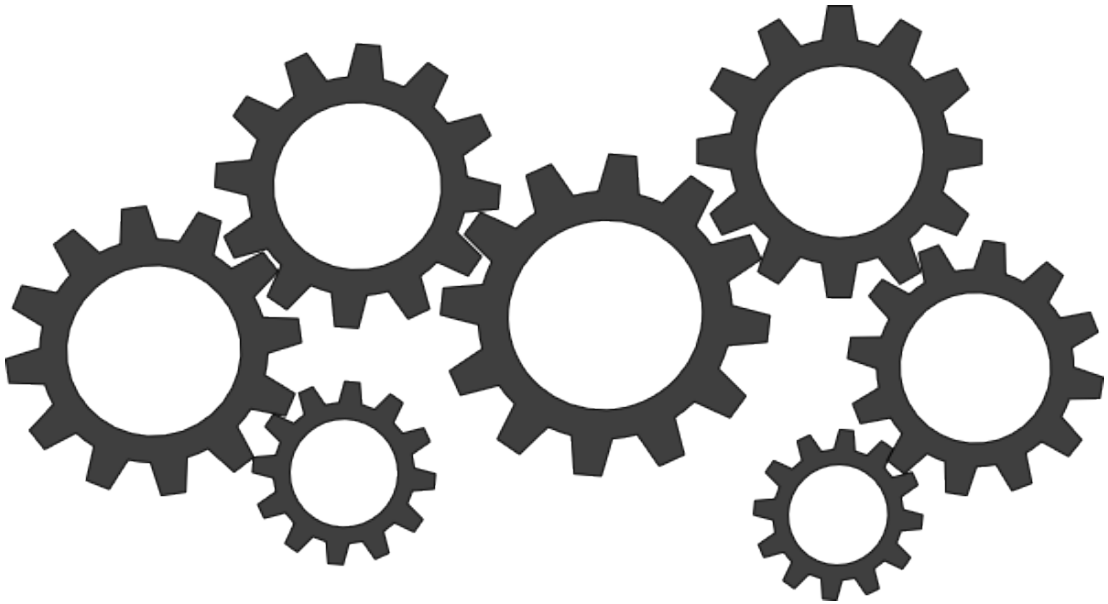
- Kada želimo da napustimo `less` dovoljno je da pritisnmo `q`
- Less je jako koristan kada ga koristimo u kombinaciji sa drugim alatima:

```
$ ls -la | less
```

nano

- Koristeći `less` mogli smo samo da čitamo sadržinu nekog fajla...
- Ali fajlove nekada želimo i menjati iz komandne linije!
- Tu na scenu stupaju tekstualni editori kao što je `nano` (i malo komplikovaniji `vim`)
- Ali o tome neki drugi put...

Demo file i text alata



sort

- Prosta komanda koja sortira redove fajla prema nekom zadatom kriterijumu.
- Neke korisne opcije:
  - `-b` – ignoriše vodeće razmake
  - `-d` – sortira alfabetski (dictionary order)
  - `-f` – nije case-sensitive
  - `-r` – invertuje redosled rezultata

find

- Kao što ime nalaže, služi za pronalaženje fajlova u filesistemu na osnovu izraza. Osnovni format je:

```
$ find location comparison-criteria search-term
```

- Neke korisne opcije:
  - `-name` – traži fajlove po imenu
  - `-iname` – traži fajlove po imenu, ali nije case-sensitive
  - `-maxdepth` – limitira dubinu pretrage
  - `-not` – negira rezultate opcije koja sledi
  - `-o` – ili operator za opcije
- Na primer:

```
$ find -maxdepth 2 -name '*.php' -o -name '*.txt'
```

vraća sve fajlove koji se završavaju na php ili na `txt` do dubine 2.

Sintaksa regularnih izraza (šablona)

Neki od meta-karaktera su:

- “`.`” – uparuje se sa bilo kojim karakterom (sem novog reda)
- “`*`” – uparuje prethodni objekat nula ili više puta
- “`+`” – uparuje prethodni objekat jednom ili više puta
- “`?`” – uparuje prethodni objekat opciono (0-1 puta)
- “`^`” – uparuje se sa početkom linije
- “`$`” – uparuje se sa krajem linije
- “`[...]`” – uparuje se sa jednim od navedenih karaktera (koristitmo “`^`” da negiramo i “`-`” za opsege)
- “`\(... \)`” – grupiše deo regexa
- “`\{n, m\}`” – uparuje n, m puta
- “`\`” – uklanja specijalno značenje narednog karaktera

grep

- Služi za pronalaženje šablona u tekstualnim fajlovima. Kada je šablon pronađen, grep vraća sadržinu odgovarajuće linije teksta. Osnovni format je:

```
$ grep "regular␣expression" filename
```

- Neke korisne opcije:
  - `-i` – šablon nije case-sensitive
  - `-v` – vraća sve linije koje ne sadrže šablon
  - `-c` – vraća samo broj linija koje sadrže šablon
  - `-n` – prikazuje i redni broj linije u fajlu
- Na primer:

```
$ grep -n "{[0-9],9}" greptest.txt
```

vraća sve linije fajla `greptest.txt` koje sadrže devetocifren broj.

diff

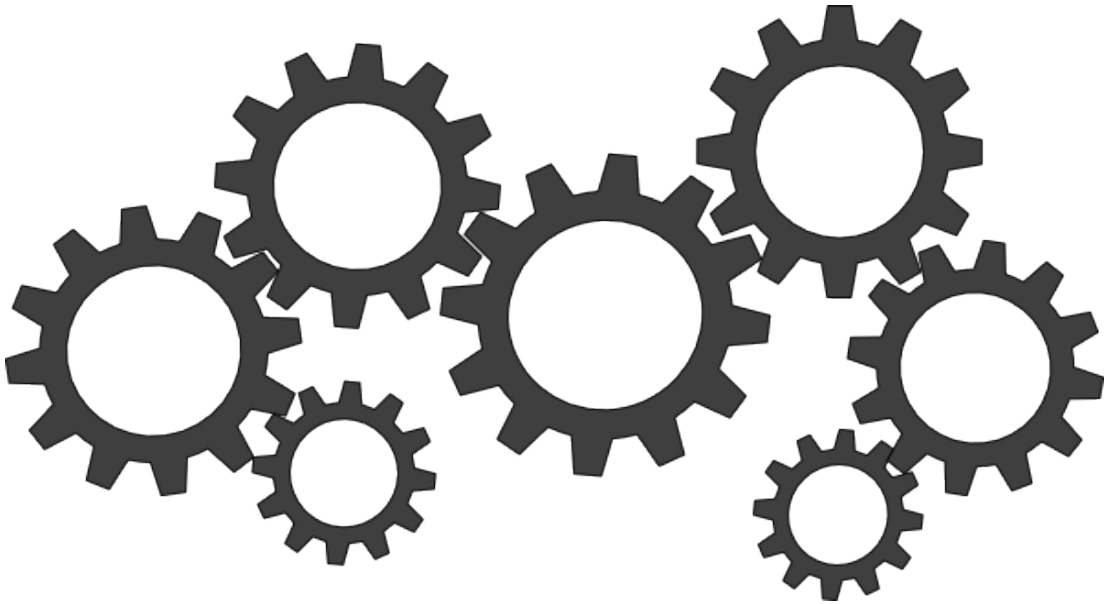
- Analizira sadržinu dva fajla i ispisuje linije koje su različite. U suštini ispisuje niz uputstava *kako promeniti prvi fajl da bude isti kao drugi.*
- Potpis komande:

```
$ diff prvifajl drugifajl
```

- Prvi red izlaza sadržaće:
  - brojeve redova prvog fajla (npr. 2,4)
  - slovo (**a** kao *add*, **c** kao *change* ili **d** kao *delete*)
  - brojeve redova drugog fajla

  Naredni redovi sadržaće redove iz oba fajla (< na početku reda označava prvi fajl a > drugi)
- Korišćenjem komande `-c` dobijamo *contextual output*.

Demo file i text alata cont'd



apt-get

- Radi sa Ubuntu *Advanced Packaging Tool* (APT), i omogućava funkcije kao što su instalacija softvera, update, upgrade i uklanjanje postojećih paketa, ali i upgrade celokunog Ubuntu sistema.
- Instalacija paketa:

```
$ sudo apt-get install imepaketa
```

- Uklanjanje paketa:

```
$ sudo apt-get remove imepaketa
```

- Update indeksa paketa:

```
$ sudo apt-get update
```

- Update svih paketa:

```
$ sudo apt-get upgrade
```

ssh

- Skraćeno od Secure SHell
- Omogućava bezbedan pristup računarima preko mreže
- Najčešća upotreba:

```
$ ssh -v username@host
```

što nam omogućava da se povežemo na host mašinu preko našeg korisničog imena, nakon što ukucamo šifru. Na primer:

```
$ ssh -v petarv@123.45.6.7
```

- Kako bi se sigurno povezali na neku mašinu, ovo nije dovoljno! Moramo zadovoljiti i sigurnosne mere:

```
$ ssh-keygen -t rsa
```

generiše naš javni i tajni ključ koristeći RSA. Da bismo se sa nekim povezali, moramo im dati naš javni ključ, a oni ga moraju lokalno sačuvati u `.ssh/authorized_keys`.

scp

- Skraćeno od Secure copy
- Omogućava bezbedno kopiranje fajlova na remote mašine.
- Najčešće upotrebe:

```
$ scp username@host:file local_directory
$ scp local_file username@host:remote_directory
```

- Na primer:

```
$ scp petarv@123.45.6.7:cope.txt /ovde
```

rsync

- Omogućava komplikovanije kopiranje i pomeranje fajlova po direktorijumima
- Najčešća upotreba:

```
$ rsync [options] source destination
```

- Na primer (vrlo pažljiv backup):

```
$ rsync -e ssh -v -rlpt --delete --backup \
  --backup-dir OLD/‘date -Im‘ \
  me@myhost.org:. mycopy/
```

  Uklanja fajlove iz `mycopy/`, koji nisu više u `me@myhost.org:.`, ali zadržava timestampovanu kopiju svakog promenjenog ili obrisanog fajla u folderu `/OLD/yyyy-mm-dd.../`.
- Ne prenosi fajlove koji se već nalaze na destinaciji. Efikasan algoritam određuje koji bajtovi treba da se prenesu, dakle vrlo je korisna metoda za backup preko sporih veza.

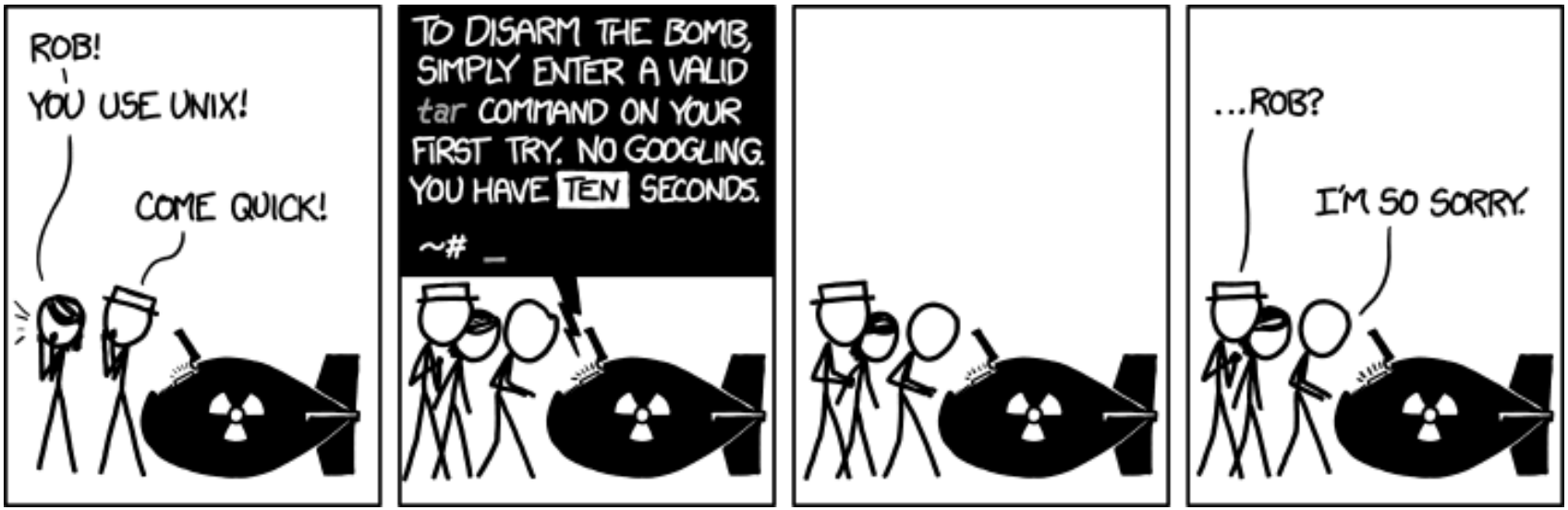
tar

- Skraćeno od Tape archiver
- Originalno napravljen u ranim danima Unixa sa ciljem backupa na magnetne trake. Moderna varijanta se koristi za distribuciju i arhiviranje fajlova, pritom čuvajući file system atribute.
- Upotreba (| je ili!):

```
$ tar [-] A --catenate --concatenate | c --create |
    d --diff --compare | --delete | r --append |
    t --list | --test-label | u --update |
    x --extract --get [options] [pathname ...]
```

- Korisne opcije:
  - `-cvf` – uncompressed (.tar) (create, verbose, file)
  - `-cvzf` – gzipped (.tar.gz) (brži i lošiji, ali se češće koristi)
  - `-jcvf` – bzipped (.tar.bz) (bolji i sporiji, koristi Burrows-Wheeler transformaciju)
  - `-xfv` – decompress (nezavisno od formata)

Tar bomb


ROB!
YOU USE UNIX!
COME QUICK!
TO DISARM THE BOMB, SIMPLY ENTER A VALID tar COMMAND ON YOUR FIRST TRY. NO GOOGLING. YOU HAVE TEN SECONDS.
~# _
...ROB?
I'M SO SORRY.

sed – stream editor

Služi za modifikaciju fajlova u jednom prolazu, i jako je koristan za automatsko editovanje teksta u pajpovima. sed skripte se mogu proslediti na komandoj liniji

```
$ sed [-e] command files
```

ili u odvojenom dokumentu

```
$ sed -f scriptfile files
```

ali uopšteni format jedne komande je

```
$ [address[,address]][!]command[arguments]
```

sed – detalji

- Adrese mogu biti redni borjevi linija ili šabloni
- Poslednja linija je “$”
- Jedna adresa bira red, dve adrese opseg (označavaju redni broj početnog i krajnjeg reda)
- Sve komande se primenjuju redom na fajl
- Nakon obrade jednog reda, on se ispisuje (sem ukoliko je korišćena opcija `-n`, u kom slučaju samo komanda `p` će ispisati liniju)
- Znak uzvika negira odabir adrese
- `sed` koristi istu sintaksu za regularne izraze kao i `grep`

sed – primeri

Menja sva pojavljivanja reči “Windows” sa “Linux” (komande: s = substitute, opcije: g = “global” = sve pojave u redu)

```
$ sed 's/Windows/Linux/g' filenames
```

Briše sve redove koji se ne završavaju sa “OK” (komanda: d = delete)

```
$ sed '/OK$/!d' filenames
```

Ispisuje linije između onih koje počinju sa BEGIN, odnosno END, inkluzivno:

```
$ sed -n '/^BEGIN/,/^END/p' filenames
```

Zamenjuje prvu reč koja počinje velikim slovom sa “X” u redovima 40-60:

```
$ sed -n '40,60s/[A-Z][a-zA-Z]*/X/' filenames
```