Naredbe za koje možda niste znali

Matko Kosmat | 1 travnja, 2026

Linux, kao uostalom i drugi Unixi, ima mnoštvo naredbi i malih specijaliziranih alata koji dolaze sa sustavom (ili u Linuxovom slučaju, sa osnovnim instalacijskim paketom). Čak i nakon duljeg bavljenja Linuxom moguće je da niste susreli neke naredbe, odnosno alate, iako bi vam mogli pomoći u svakodnevnom održavanju suatava.

Iz tog razloga, pokušat ćemo napraviti pregled nekih alata koje smo imali priliku susresti, a nisu (toliko) uobičajeni. Podrazumijeva se da popis nije, niti ikada može biti kompletiran, no svakako ćemo probati pronaći i opisati najzanimljivije i najkorisnije alate.

Na Linuxu, kao derivatu Unixa, ali i na svakom drugom operativnom sustavu može doći do problema i grešaka. Najčešće su to jednostavniji problemi koje možete samostalno riješiti, jer je opis greške dovoljno jasan da se može vidjeti u čemu je problem. Problem nastaje u trenutku kad je greška previše kriptična, previše dugačka ili jednostavno nerazumljiva. Tu u pomoć može uskočiti naredba "script".

Naredbom script dižete novu korisničku ljusku (*shell*), ali s tom razlikom da će se sve što se događa u njoj bilježiti u datoteku "typescript" u trenutnom direktoriju, odnosno direktoriju u kojem ste pokrenuli naredbu. Na ovaj način imate detaljan pregled što ste radili, i što se potom događalo.

Script ima manu da ne radi dobro s programima koji manipuliraju ekranom na bilo koji način (pine, mutt, lynx...), odnosno bit će teže protumačiti što se događalo. No, i dalje su to vrlo korisne informacije i ne treba izbjegavati script zbog toga.

Jedna od najkorisnijih, naredba apt-get može često izbaciti pitanje ili grešku na koje ne znate odgovor, pa morate potražiti pomoć nekog drugog, ili pomoć odgovarajuće službe. Način uporabe je jednostavan, samo treba pokrenuti naredbu script, pa onda već uobičajeni niz apt-get update & upgrade ili nešto drugo:

Dakle, uspjeli smo sačuvati sav izlaz naredbe apt-get, kao i vaše akcije, što će umnogome pomoći kod rješavanja problema ako datoteku typescript pošaljete kao prilog poruci. Bez ovih informacija, vaš problem će biti znatno teže i sporije riješen.

Ime izlazne datoteke ne mora biti typscript, može biti bilo koje drugo ime:

Ostale, malobrojne, opcije možete naći u kratkom manualu naredbe, koji ćete dobiti već poznatom "man script" naredbom.

Svima je jasno da je IP prostor sve manji, te da institucije dobivaju sve manje IP adresa. Jedan od načina rješavanja problema nedostatka IPv4 prostora je i bezklasno adresiranje (Classless Inter-Domain Routing) - CIDR.

Ovdje vas nećemo pokušavati naučiti što je CIDR, nego to ostavljamo vama. Dobar početak je Wikipedia, koja ima unos o CIDR-u na URL-u: http://en.wikipedia.org/wiki/Classless_Inter-Domain_Routing

Cilj članka je olakšati rad sa CIDR zapisima, koji su obično u obliku

primjerice

Dakle, ovo je lako riješiti, jer /24 znači da je 24 bita upotrijebljeno za adresiranje mreže, dok je ostatak namijenjen za hostove.

Ovo je standardno podešavanje za većinu institucija u CARNetu, i dodjeljuje 256 adresa na uporabu instituciji. No, što ako imamo slučaj

Ovo se da izračunati ako adresu pretvorimo u binarni oblik ili pogledamo u kakvu tablicu. No, još

lakše je upotrijebiti alat ipcalc:

Sad je sve jasno i bez ikakvog računanja. Maska /28 zapravo znači da je moguće adresirati samo 14 hostova (16 - 2 zbog mrežne i broadcast adrese), te da je raspoloživi raspon adresa od 193.198.1.1 do 193.198.1.14.

Moguće je i obrnuto, kako saznati CIDR ako imamo uobičajene podatke o IP adresi i mrežnoj masci? Poslužimo se naredbom "ifconfig":

Vidimo dakle IP adresu, broadcast adresu i mrežnu masku. Napravimo:

Možemo vidjeti iste podatke kao i u prethodnom primjeru, ali i CIDR oznaku, koja u konkretnom slučaju glasi 193.198.1.64/26.

Naredba ipcalc nam, dakle, omogućuje brzi i nepogrešiv prikaz raspona adresa u bilo kojem obliku. Ipcalc ne prima mnogo opcija, a spomenut ćemo samo neke najkorisnije:

-n ne prikazuje boje koje su po defaultu uključene
-b ne prikazuje binarni raspis adresa
-h ispis je u HTML obliku

Više informacija možete pročitati u manualu naredbe ("man ipcalc").

U seriji članaka pod nazivom "Naredbe za koje (možda) niste znali" pokušat ćemo vas upoznati s nekim naredbama i dodatnim priručnim alatima koji se ili standardno nalaze na Unix i Linux sustavima, ili se često dodaju zbog svoje uporabnosti i popularnosti. S jednim takvim alatom smo vas već upoznali (ipcalc), a ovim člankom počinjemo novi ciklus.

Prvi alat s kojim ćemo vas upoznati je "units". Kao što mu samo ime govori, radi se o praktičnom alatu za pretvorbu raznih mjernih jedinica u naredbenoj liniji.

Pretvorba se može raditi interaktivno ili neinteraktivno, primjerice ukoliko želimo saznati koliko litara ima u jednom galonu:

Izlazak iz interaktivnog načina rada je moguć uporabom standardne kombinacije tipki Ctrl+C. Koji ćete oblik rabiti, ovisi o Vama.

Malo ćemo objasniti ispis. Zvjezdica ispred vrijednosti znači da je vrijednost pretvorbe dobivena linearno, množenjem sa nekim koeficijentom. Iza zvjezdice je, naravno, vrijednost u traženim jedinicama. Druga linija određuje obrnuti koeficijent, s kojim možemo dobiti originalnu mjernu jedinicu (1 litra je 0.264 * 1 galona).

Za razliku od linearnih, postoje i nelinearne pretvorbe, kao što je primjerice pretvorba iz stupnjeva Fahrenheita u stupnjeve Celzija.

Ovaj put, dakako, nema zvjezdice, a treba uočiti i drugačiji način pisanja vrijednosti za temperaturu, koja se piše slično pozivanju funkcije u C-u. Interaktivno, ne trebaju navodnici koji štite da ljuska ne interpretira zagrade:

Neke pretvorbe nisu očigledne, kao recimo pretvorba iz milja u kilometre:

Ukoliko dobijete poruku "conformability error", znači da jedinice nisu uskladive, i ispisat će se njihove standardizirane vrijednosti (50 milja na sat je oko 22 metra u sekundi, i definicija jedinice koju program nije razumio). Ispravan način definicije kilometra na sat nije ni "km/h", ni "kmh", nego "kph":

Još sigurnije je koristiti govorni oblik "kilometers per hour", pri čemu britanski i američki način pisanja ne igraju ulogu (mogli smo napisati i "kilometres").

Ukoliko samo želite vidjeti definiciju neke jedinice, primjerice koliko je to 1 tekuća unca:

Dakle, 1 tekuća unca je 0.3 decilitara.

Alat units podržava mnogo više od ovih uobičajenih mjernih jedinica, pa i neke arhaične i napuštene.

Svakako preporučujemo da pogledate man stranicu i datoteku s pretvorbama /usr/share/misc/units.dat, jer ćete tamo naći mnoštvo informacija koje će vam zatrebati, poglavito zbog pravilnog pisanja mjernih jedinica i njihovih parametara.

Ovaj zgodni alat možete instalirati, kao i uvijek, preko apt-get naredbe:

Ime naredbe cal dolazi od engleskog "calendar", što automatski otkriva i njenu namjenu. Pomoću naredbe cal možete ispisati, obraditi i potom isprintati kalendar praktički bilo koje godine, uključujući i one po Julijanskom kalendaru.

Raspon godina koji cal može prikazati je od 1. do 5875706. godine, dakle, nećete ovaj alat tako skoro prestati rabiti.

Najčešća primjena je ispis trenutnog mjeseca:

Naredba prima jedan ili dva parametra. Ukoliko ste naveli jedan parametar, smatra se da ste mislili na godinu. Kod navođenja godine valja obratiti pozornost da morate otkucati cijelu godinu, dakle "cal 09" će prikazati kalendar za 9. godinu nove ere. Ispravan je oblik "cal 2009":

Ukoliko želite saznati kako će izgledati neki mjesec određene godine navedite mjesec prije godine:

Ako pogledate ispis, vidjet ćete da tjedan počinje sa nedjeljom, što nije uobičajeno u našem području. Ispis kalendara s ponedjeljkom kao prvim danom dobit ćete primjenom opcije "-m":

UPDATE 03-2011: U izdanju "Squeeze" Debiana morat ćete rabiti drugačiji oblik naredbe da biste dobili ponedjeljak kao prvi dan u tjednu, **ncal -M**:

Još jedna zanimljiva opcija je "-3". Ona će prikazati mjesec koji ste tražili, ali i mjesec prije i mjesec poslije, sve u jednom redu. Ovo može dobro poslužiti za planiranje godišnjih odmora, primjerice.

Osim naredbe cal, postoji i emulacija naredbe ncal, koja se prvo pojavila na FreeBSD sustavima. Kad cal pozivate kao ncal, ispis je ponešto drugačiji, kompaktniji i tjedan počinje s ponedjeljkom:

Kao zanimljivost, navest ćemo da ncal podržava prikaz dana kada padaju Uskršnji blagdani, samo trebate upotrijebiti opciju "-e" (od engleskog *Easter*):

Dakle, iako nemaju previše opcija, naredbe cal i ncal nam ipak mogu pružiti brze, točne i korisne informacije. Kalendare kasnije možemo formatirati i isprintati u nekom programu za obradu teksta ili softveru za DTP, ukoliko nam pri ruci nije neki "pravi" kalendar.

Cal i ncal se nalaze u paketu "bsdmainutils", pa se jednostavno daju instalirati sa:

ukoliko ih već nemate na sustavu.

Svima nam poznata operacija kojoj obično prethodi copy. Ali u linux komandnoj liniji naredba paste postaje nešto sasvim drugo.

Naredba paste služi za spajanje redaka iz dvaju ili više datoteka. Primjerice, imamo tri datoteke. U prvoj se nalaze imena ljudi, u drugoj telefonski brojevi, a u trećoj mjesta.

Datoteka 1 – dat1.txt

Datoteka 2 – dat2.txt

Datoteka 3 – dat3.txt

Izvršavanjem naredbe

dobijemo datoteku dat.txt sljedećeg sadržaja:

Ukoliko želite datoteke složiti po stupcima dovoljno je uporabiti parametar -s i kao rezultat dobije se:

Možda nam se ovakav način obrade čini pomalo arhaičan, ali poznavanje ovakvih naredbi neki put zaista dobro dođe. Primjerice, imamo datoteku s korisničkim imenima, a od nas se traži da ju pripremimo za unos u program za automatizirano dodavanje korisnika, koji kao ulaz očekuje datoteku u formatu korisnik:lozinka.

Datoteka korisnici.txt:

Programom za generiranje lozinki izgeneriramo potrebne lozinke i snimimo ih u datoteku lozinke.txt.

Naredbom

dobijemo datoteku za unos korisnika u traženom formatu.

Naredba **cut** u linuxu je komplementarna naredbi **paste** i s njom se mogu na brzinu napraviti određene manipulacije s tekstualnim datotekama, za koje bi možda inače trebali pisati perl skriptu ili koristiti **awk**.

Kako bi vidjeli što ta naredba uistinu radi, najbolje je da pokažemo na jednom primjeru.

Prvo napravimo jednu probnu datoteku **proba.txt**. Za pravljenje **proba.txt** datoteke ne moramo posezati za editorom, nego, kao iskusni sistemci (zlo)upotrijebimo konzolnu naredbu **cat** i makro naredbu **EOF** (End Of File), te upisujmo kako slijedi:

(za ramake između stavki koristili smo tabulator).

Pogledajmo jel' datoteka **proba.txt** kreirana i kako izgleda:

Dobili smo tekstualnu datoteku, sastavljenu od redova i stupaca. S naredbom **cut** možemo "isjeći" bilo koji stupac, ili stupce, recimo drugi i četvrti stupac:

*(Redoslijed stupaca uvijek ostaje, bez obzira, kako smo poredali parametre iza **-f** (field) opcije, pa bi tako rezultat bio isti i da smo gornju naredbu pisali: **cut -f 4,2 proba.txt**. Drugi red u datoteci **proba.txt** je namjerno ostavljen prazan, da se vidi da to ne utječe na korektno izvršavanje naredbe)*

U navedenom primjeru tekstualna datoteka je imala tabulator kao *delimiter*, no ako se između pojedinih stupaca nalazi neki drugi znak, npr dvotočka: **:** ili razmak **' '**, onda se uz opciju **-f** treba dodati opcija **-d':'** odnosno **-d' '**. Ako trebamo "isjeći" redove umjesto stupaca, onda trebamo predhodno upotrijebiti komandu **paste** s parametrom **-s**, i nakon primjene comande **cut**, opet primijeniti komandu **paste** s parametrom **-s** da opet budu redovi - redovi, a stupci - stupci. S **cut** i **paste** vrlo je zgodno koristiti i **sort**, da bi se redovi i stupci sortirali po određenom kriteriju - npr. po abecedi ili rednom broju.

Za ovih pet naredbi smo odlučili napisati jedan članak, jer su pojedinačno prejednostavne i imaju premalo funkcija, a opet su na jedan način povezane. Ovi programi nude nekoliko načina pretrage datoteka, naredbi ili stranica manuala za bilo koju naredbu koja vam je u tom trenutku potrebna.

Ukoliko želite pronaći datoteku ili direktorij (nije bitno sjećate se imena cijele naredbe, dovoljno je znati samo dio naziva te naredbe), možete upotrijebiti naredbu **locate**:

locate

Naredba **locate** pronalazi sve datoteke i direktorije koje u svom imenu imaju niz znakova '**peri**'. Naredba rabi malu bazu koja se (obično, zanemarit ćemo ovaj put *triggere*) generira svaki dan putem crona (konkretna skripta je /etc/cron.daily/find). Sama baza se nalazi u direktoriju /var/cache/locate/locatedb, a generira se putem naredbe 'updatedb'.

Iako će biti u većini slučajeva obnova baze jednom dnevno biti sasvim zadovoljavajuća, mana naredbe locate je nesinhroniziranost sa stvarnim stanjem. Iz tog razloga, bazu možete ručno osvježiti, preko već spomenutih naredbi i skripti:

ili

Osvježavanje traje neko vrijeme (naravno, ovisi o brzini poslužitelja), pa budite strpljivi. Konfiguraciju možete vidjeti u /etc/updatedb.conf, iako nije vjerojatno da ćete tamo tamo trebati išta mijenjati. Za potpune informacije pogledajte stranice manula za naredbe **locate**, **updatedb** i **locatedb**.

whereis

Ukoliko tražite specifično samo naredbu (ili njen izvorni kod) i odgovarajuću stranicu manuala (dakle, neće biti ispisane baš sve datoteke koje odgovaraju upitu), pomoć nudi naredba '**whereis**':

Naredba ponekad može dati i nepredvidljive rezultate (zbog *hardcodiranih* vrijednosti unutar programa), ali su i dalje upotrebljivi:

Naredba **whereis**, za razliku od ostalih sličnih naredbi, ***ne pretražuje pomoću baze***, stoga su podaci koje ispisuje aktualni i usklađeni sa trenutnim stanjem paketa na sustavu. Pretragu možete dalje ograničiti putem opcija -b (traži samo binarne, odnsono izvršne datoteke), -s (traži samo datoteke izvornog koda) i -m (traži samo dateoteke stranica manuala). Primjer:

whatis

Tradicionalno, **whatis** se mogla pozivati i preko oblika 'man -f'. Za razliku od naredbe sličnog imena **whereis**, **whatis** pretražuje samo u zaglavljima stranica manuala. Zaglavlje stranice manuala, kao što vam je poznato, izgleda otprilike ovako:

Dakle,ukoliko na brzinu želite znati što koja naredba radi, na naredbenu liniju ukucat ćete ovo:

Naredba **whatis** traži samo cijelu riječ (nema skraćivanje):

No, uporabom opcije '-r' (od *regex,* regularni izraz), možemo proširiti popis rezultata:

apropos

Tradicionalno, ova naredba se rabila kao "man -k".

Naredba **apropos**, za razliku od whatis, pretražuje cijelo zaglavlje stranice manuala, pa će prikazati sve stranice manuala na sustavu koje u zaglavlju spominju traženu riječ. Ta riječ se uopće ne mora pojavljivati u nazivu programa!

Na ovaj način možete u jednom potezu pronaći sve relevantne stranice za neki program, bez gledanja u pojedinačne stranice manuala.

Naredbe **whatis** i **apropos**, kao i **locate**, ispisuju rezultate pomoću indeksirane baze, ali to nisu iste baze. U Debianu se baza za **whatis** i **apropos** nalazi u datoteci /var/cache/man/index.db.

Ovo znaèi da baza mora biti osvježena kako bi vaše pretraživanje dalo relevantne rezultate. O ovome se brine cron datoteka, ali slično kao i za **locate** bazu, indeksiranje možete pokrenuti i ručno:

which

Zadnja s prethodnim naredbama srodna (po nama), je naredba **which**, koja ne traži ništa po datotečnom sustavu (jednostavno, pretražuje trenutnu stazu, $PATH), nego ispisuje koja će se naredba izvršiti ukoliko je otkucate na naredbenoj liniji:

Naredba je korisna ukoliko imate dvije ili više naredbi istog imena, ili želite provjeriti je li naredba uistinu ona za koju mislite da će se pokrenuti u trenutnoj stazi ($PATH-u). Ovo je korisno kad postoje dvije ili više naredbi istog imena (klasičan primjer je naredba "**host**").

Problem: Izlistate direktorij i u imenima datoteka vidite osim *normalnih* i razne čudne znakove, npr. kvadratiće, razne grafičke simbole, prazne prostore - ili najčešće upitnike. Preimenovati takve datoteke pojedinačno, vrlo je sporo, a standardni alati koji to obavljaju nad grupama datoteka, ne mogu samo tako pomoći, jer je sustav imena preheterogen i često se ne da obuhvatiti nekim pravilom.

Odmah vam pada na pamet, kako bi zgodno bilo cijeli direktorij editirati, kao tekstualnu datoteku - ali čime?

Na sreću, rješenje postoji. Tu je naredba **vidir**.

Naredba **vidir** u sadašnjoj CARNet distribuciji Debiana ne dolazi standardno ugrađena, ali se lako dogradi uz ostale korisne alate iz paketa **moreutils** naredbom **apt** iz konzole:

Sada je dovoljno da odete u direktorij s datotekama kojima želite editirati imena i upišete:

(ako ne želite editirati i imena poddirektorija možete ispustiti minus '**-**' *)*

Pokrenut će se, suprotno očekivanju, vaš podrazumijevani editor (a ne nužno **vi**, što bi sugeriralo ime **vi***dir*, tj. onaj isti koji se pokrene kad pokrenete recimo naredbu **crontab -e**) i dobit ćete unutar njega cijeli direktorij (samo imena datoteka) u obliku jedne nove tekstualne datoteke, koju možete ispravljati i korigirati po želji. Nakon pospremanja "datoteke" i pokretanja naredbe **ls** promjene će biti odmah vidljive.

Usput napomenimo, da u Debianu vrlo lako možete zamijeniti podrazumijevani editor naredbom **update-alternatives --config editor**:

Još samo jedno upozorenje za svaki slučaj: ako se s **vidir** poslužimo bez puno eksperimentiranja i nepotrebnog "igranja", vidjet ćemo da je to zbilja koristan alat, u protivnom bi mogli.ostati bez poneke datoteke.

Naredba file(1) je jedna od korisnijih naredbi za brzo utvrđivanje tipa datoteke. Unix sustavi, uključujući i Linux, nemaju posebne ekstenzije za datoteke koji bi određivali tip datoteke (.txt, .doc). Kako bi bez otvaranja datoteke u nekom editoru (koji inače služe samo za pregled tekstualnih datoteka) saznali o kakavoj se datoteci radi, možemo se poslužiti naredbom **file**:

Saznali smo da je datoteka /bin/ps 32-bitna izvršna binarna datoteka, kompatibilna sa x86 arhitekturo. Dodatno, program rabi dinamične biblioteke (DLL-ove), te su iz nje maknuti svi simboli iz kompiliranog koda (programeri će znati o čemu je riječ, a za ostale je dovoljno reći da je *stripana* datoteka znatno manja, a nema bez gubitka funkcionalnosti).

Datoteka sid.bmp je obična sličica u Bitmap formatu, a odmah su dane i dimenzije, kao i dubina boja (2^4 = 16 boja).

Datoteka /etc/motd je, očekivano,obična tektualna datoteka (svi znakovi su ispod ASCII 127). Detekcija jezika, isto tako, nije savršena, pa se u nju nemojte pouzdati.

Naredba file može detektirati i datoteke koje to zapravo nisu. Ovdje smo testirali datoteku koja predstavlja drugu particiju na prvom disku, a rezultat je očekivan: riječ je o posebnoj *block-oriented* datoteci, kao što su to svi diskovi. Brojevi u zagradi značе koji su *major* i *minor* brojevi te posebnedatoteke. Za ilustraciju, pogledajte i rezultat za trakovni uređaj:

Traka je, naravno, *character-based* uređaj, što naredba file uredno detektira.

Dakle, naredba file pokušava što opširnije opisati o kakvoj je datoteci riječ. To pokušava na tri načina: putem datotečnog sustava i sistemskog poziva stat(2), "magičnih" brojeva i na kraju pregledom sadržaja datoteke. Magični broj se nalazi u gotovo svakoj binarnoj datoteci, približno na početku. Popis ovih magičnih brojeva se nalazi u direktoriju /usr/share/misc/file/ u nekoliko datoteka, koje ne trebate nikad dirati jer se periodično osvježavaju u novim inačicama paketa.

Ukoliko tip datoteke nije prepoznat, možete sami upisati magični broj, ali u datoteku /etc/magic. Format upisa možete saznati sa "man magic".

Naredba **diff** je jedna od onih naredbi za koje nismo bili sigurni jesu li uopće pogodni za kategoriju "naredbe za koje niste znali". No, ako uzmemo u obzir da je jedan dio sistemaca uopće ne rabi, a drugi dio ne rabi sve mogućnosti, ipak je zaslužila svoje mjesto unutar ove kategorije.

Ukratko rečeno, naredba diff pokazuje razlike između dvije datoteke. Primjera radi, recimo da imamo dvije datoteke sljedećeg sadržaja:

Pogledajmo što će biti kada upogonimo naredbu diff (datoteka1 se smatra "starom", a datoteka2 "novom"):

Ovo je takozvani "normalni" izlaz naredbe diff. Oznake "manje od" i "veće od" označavaju o kojoj se datoteci radi. Cijeli ispis je zamišljen tako da opisuje što treba napraviti da "stara" datoteka postane "nova". Oznake iznad linija opisuju promjene koje treba napraviti: **a** (*add*) je novododana

linija, **d** (*deleted*) je obrisana linija, a **c** (*changed*) je promijenjena linija. Brojevi označavaju redni broj linije, s time da je broj koji se nalazi ispred slova broj linije u staroj datoteci, a broj iza slova broj linije u novoj datoteci.

Protumačimo te oznake:

- u staroj datoteci prva linija je "aaaaaaaaa", koju treba obrisati (stoga stoji nula u novoj datoteci)
- u staru datoteku na treću poziciju treba dodati "ddddddddd", koja je na trećoj poziciji i u novoj datoteci

Linije koje su zajedničke objema datotekama se po defaultu ne ispisuju.

No, format koji se najviše rabi nije ovaj, normalni, nego kontekstualni. Kod je najbitnija činjenica to da ne rabi brojeve linija, nego promjene smješta u kontekst originalne datoteke. Ovo postiže tako da uključi nekoliko nepromijenjenih linija na početku i na kraju promjena (obično 3 linije). Tako se kreiraju tzv. *hunkovi* ili *chunkovi*, dijelovi promjena nad istom
datotekom. Ovakav način je efikasniji od uobičajenog, te je datoteka manja nego drugi formati (naziva se "patch" datoteka).

Ovakav izlaz možemo dobiti s opcijom **-c** (od "*context*"), no puno je uobičajeniji "unified context" format, kojeg dobijemo s opcijom **-u**. Upravo ovakav ispis daje Debianov paketni sustav, kad odaberete opciju "**D**" kada vas sustava pita želite li zadržati staru ili prihvatiti originalnu konfiguracijsku datoteku paketa. *Ovo je ujedno i glavni razlog zašto inzistiramo na unified formatu, i zašto uopće spominjemo naredbu diff.*

Najlakše je to razumjeti na primjeru:

Prve dvije linije su gotovo identične, a najbitnija je razlika ta što je originalna datoteka označena s "---", a nova s "+++". Sljedeća oznaka je "@@ -1,13 +1,11 @@". Ovo označava da počinje *chunk*, te koliko linija obuhvaća. Dvostruki @ simboli su samo oznaka početka i kraja informacija o chunku. Prva skupina brojeva, označena minusom, se odnosi na originalnu datoteku, te označava početak i na koliko linija se odnose promjene (počinje od prve linije, i završava s trinaestom). Ista formula se primjenjuje na novu datoteku, konkretno promjena počinje s prvom linijom, i završava s jedanaestom.

Sam *chunk* je označen na jedan od tri načina: bez oznake, sa znakom "+" ili sa znakom "-". Slično kao i prije, ukoliko linija nema oznaku, znači da nema ni promjene. Ukoliko je označena sa "+", znači da se ta linija dodaje i suprotno, ukoliko je označena sa "-", ta linija se briše. Vratimo se originalnom primjeru:

Na ovom primjeru možemo vidjeti da se promjene u obje datoteke provode na prve tri linije, da se linija "aaaaaaaaa" mora obrisati, a linija "ddddddddd" dodati kako bi se od stare napravila nova datoteka.

To nije sve, diff može uspoređivati i direktorije. Kreirali smo direktorije (analogno sadržaju datoteka):

Rezultat ove naredbe je, nadamo se, svakome jasan.

Kao kuriozitet, moramo spomenuti i naredbu **diff3**. Ako iz imena nije jasno, diff3 uspoređuje i ispisuje razlike u čak tri datoteke. Ukoliko se pitate čemu to može poslužiti, reći ćemo da je glavna namjena diff3 ujedinjavanje promjena (*merge*) koje su dvije osobe napravile nad istom datotekom (npr. neki izvorni kod). Kako ovo već ulazi u područje raznih softvera za nadzor revizija, vjerojatno ćete takvo nešto i rabiti umjesto diff3.

Ipak, ako trebate istodobnu usporedbu tri datoteke, sada znate da i to možete. Ograničit ćemo se, doduše, samo na ispis naredbe diff3, a tumačenje ostavljamo za domaću zadaću (pametniji će odmah pokrenuti man diff3):

Za usporedbu datoteka imamo na raspolaganju još kandidata. Osim moćne naredbe diff, postoje još dvije naredbe koje ćete moći upotrijebiti u svakodnevnom radu. Prva od njih je **comm**.

Comm je nešto skromnijih mogućnosti, i rabi se najviše u skriptama, ali i za brzi vizualni pregled razlika između datoteka. I izlaz ove naredbe je zanimljiv:

Izlaz naredbe je u tri stupca. U prvom stupcu se nalaze linije jedinstvene za datoteku 1. Srednji stupac sadržava linije jedinstvene za datoteku 2. I na kraju, zadnji stupac sadržava linije koju su zajedničke za obje datoteke. Bilo koji od ovih stupaca se može isključiti, ukoliko tako želite (upotrijebite opcije -1, -2 i -3).

To je uglavnom sve što ova naredba nudi. Na kraju, spomenut ćemo naredbu **cmp**. Ona je naprikladnija za uporabu u skriptama, dok za vizualni nije previše upotrebljiva. Radi na principu uspoređivanja bajt po bajt:

Dakle, prijavio je da se datoteke razliku odmah u prvom bajtu, što nam i nije neka velika pomoć. No, zato ćemo promatrati njezinu izlaznu vrijednost, kako bismo je mogli upotrijebiti u skriptama:

Dakle, naredili smo da cmp ne ispisuje ništa (preko opcije **-s**, *silent*), a zatim smo samo provjerili izlaznu vrijednost. Izlazna vrijednost je 1 ako se datoteke razlikuju, a 0 ako su identične (i 3 ako je došlo do nekakve greške). To se lako može iskoristiti u skriptama, navest ćemo primjer uporabe:

To je bilo sve što smo predvidjeli napisati za ove naredbe, na vama je sad da gore opisane naredbe upotrijebite na (vama) koristan način. U sljedećem dijelu opisat ćemo naredbe komplementarne ovima: patch i merge.

Kao i svi operativni sustavi, tako i linux s vremenom nakuplja sve više nepotrebnog softvera, što nakon nekog vremena počinje produljivati razne uobičajene operacije na računalu (npr. backup), i jednostavno nepotrebno zauzimati prostor na disku. Ovo nije problem operativnog sustava, nego operatera, a to ste vi.

Svaki program, odnosno paket, koji instalirate na Debian, zahtijeva dodatne biblioteke (Windowsaši bi rekli DLL-ove). Neki su skromni, i ne treba im ništa drugo osim osnovne (g)libc biblioteke. Neki, posebice programi pisani u interpretiranim jezicima poput perla, zahtijevaju instalaciju 5, 10 pa čak i više dodatnih paketa kako bi mogao funkcionirati. Sve je to lijepo, ali suprotno se ne događa: ukoliko maknete neki takav program, neće automatski biti maknuti i svi drugi paketi na koje se taj paket oslanja (**Depends**). Primjerice:

Nakon nekog vremena shvatite da taj program ne radi ono što želite, i kao svaki savjestan sistemac, obrišete ga sa sustava:

To će obrisati paket '**paket**', ali će nezainteresirano ostaviti **paket-client** i **paket-doc** na sustavu. To pomnožite s godinama i u nekoliko se godina može nakupiti nekoliko desetaka takvih paketa, što više nije zanemariv broj (niti prostor na disku, odnosno traci).

U pomoć priskače mali program **deborphan**. On će nakon pokretanja provjeriti sve pakete o kojima više ne ovisi ni jedan drugi paket. Kako je sasvim realno da imate pakete o kojima nitko ne ovisi, a svejedno vam trebaju, deborphan će selekciju napraviti nadasve oprezno. Da vidimo kako:

Ovo je skraćeni ispis, što znači da je nepotrebnih paketa moguće imati znatno više. Pažljiviji će čitatelji uočiti da se radi isključivo o paketima biblioteka (i to najčešće u više inačica). Ovo je razumljivo, jer ako ni jedan paket ne treba određenu biblioteku, što će ona više na sustavu? Osim u posebnim slučajevima, ili ste developer (a tada vam trebaju i -dev paketi), ovi se paketi bez imalo grižnje savjesti mogu obrisati:

Kako biblioteke mogu ovisiti o drugim bibliotekama, moguće je da će se brisanjem jedne biblioteke, kao "siroče" (*orphan*) pojaviti neka druga biblioteka! Rješenje je jednostavno, nakon brisanja provjerite je li se pojavilo što novo, i ponovite postupak sve dok naredba deborphan nema ništa više za prijaviti:

Dakako, to nije sve. Deborphan podrazumijevano prikazuje samo pakete biblioteka, no s opcijom **-a** prikazuje baš sve pakete o kojima nitko ne ovisi. Ukoliko niste sigurni u to što radite, **ovdje se zaustavite**.

Postoje mnogi paketi koji ne ovise o nikome, osim o osnovnim bibliotekama sustava. Dakle, ukoliko niste **zaista** sigurni da vam taj program/paket ne treba, nemojte ga brisati.

Ukoliko ipak želite dodatno očistiti sustav, imate na raspolaganju opciju **--guess**. Ona će pokušati ograničiti potragu na određene dijelove Debian paketnih sekcija, poput **dev**, **doc**, **data** ili **dbg** (naravno, podržane su i druge sekcije, što možete vidjeti sa **man deborphan**).

Pretpostavit ćemo da niste developer, pa vam ne trebaju **-dev** paketi:

Dakle, iako nisu biblioteke, izolirali smo još neke pakete o kojima nijedan drugi paket ne ovisi, niti vama trebaju, pa ih možemo obrisati. Slično je i sa **-doc** paketima, jer dokumentaciju uvijek možete naći na webu, a i ostaju vam man stranice (one ne dolaze u **-doc** paketima, koji su dodatna dokumentacija u primjerice HTML-u ili PDF-u). Dakako, ovo sve ovisi o vama i vašim potrebama.

Paket deborphan je samostalan paket, obično se ne nalazi na sustavu i potrebno ga je instalirati sa:

Naredbu **touch** ste vjerojatno vidjeli u nekim skriptama, ali vas nije zanimalo što radi (ili vam jednostavno nije bilo jasno što bi ta naredba trebala raditi). Naredba touch ima, ukratko rečeno, dvije funkcije: može kreirati praznu datoteku, ili promijeniti vrijeme kreiranja bilo koje datoteke. Za što nam ta funkcionalnost uopće treba?

Kreiranje prazne datoteke ćemo najčešće rabiti kada neki program zahtijeva da određena datoteka već postoji, makar bila prazna. Tipičan primjer su daemoni, koji se često uopće neće pokrenuti, ili neće ništa zapisivati u svoje logove ukoliko njihova vlastita log datoteka ne postoji. Tada ćemo jednostavno kreirati log datoteku sa:

U ovakvim slučajevima će možda biti potrebno promijeniti i vlasništvo nad datotekom pomoću naredbe **chown** i **chgrp**. Moguće je odmah kreirati i više datoteka:

Drugi razlog zašto bi htjeli kreirati praznu datoteku se može naći u skriptama, koje na ovaj način kreiraju vremenske referentne datoteke (*timestamp* datoteke). Na ovaj način skripta može provjeriti, primjerice, koliko dugo se izvršava, može preživjeti restart poslužitelja ili vlastito "rušenje", što nije slučaj s internim načinom računanja vremenskih perioda.

Vrijeme kreirane datoteke možemo, osim s naredbom **ls**, provjeriti pomoću naredbe **stat**.

U slučaju potrebe, moguće je promijeniti originalne podatke o vremenu kreiranja datoteke:

Kao što se može vidjeti, vrijeme je promijenjeno na vrijeme koje je bilo u trenutku izvršavanja naredbe. No, vrijeme se može postaviti na proizvoljnu vrijednost:

Datum smo postavili na 31. prosinca 2000, a vrijeme točno u 23 sata, 59 minuta i 59 sekundi. Format upisa vremena je [[CC]YY]MMDDhhmm[.ss], dakle moguće je rabiti samo kraći oblik mjesec:dan:sat:sekunda.

Također, možemo selektivno odabrati koje ćemo vrijeme mijenjati, sa opcijom **-a** mijenjamo samo ATIME (*access time*, ako je omogućen na particiji), a s opcijom **-m** mijenjamo samo MTIME (*modification time*):

Promijenili smo samo vrijeme pristupa datoteci (ATIME možemo vidjeti s naredbom **ls -lu**, dok je MTIME vidljiv već sa standardnim **ls -l**). Promijenimo sada samo MTIME vrijeme:

Možemo vidjeti da je sada MTIME vrijeme kasnije od ATIME vremena.

Naredba touch omogućava da novo vrijeme datoteke postavite po već postojećoj datoteci, primjerice:

Opcijom **-r** smo postavili smo ATIME i MTIME po datoteci /etc/hosts (mogli smo naravno, odabrati bilo koju drugu datoteku).

Što se tiče naredbe touch, ovo bi bilo uglavnom sve, no spomenut ćemo još i opciju **-d**, koja će biti lakša korisnicima s engleskog govornog područja, jer prima drugačiji format datuma:

Pretpostavljamo da će se većina vas ipak držati "čistog" numeričkog načina zadavanja formata datuma.

Naredba touch je dio paketa **coreutils**.

Naredba **stat** nam omogućava dobijanje informacija o datotekama, kao što su veličina, vrijeme kreiranja, promjene i pristupa, inode broja i slično. Sve ovo možete saznati i preko naredbe **ls**, ali stat daje sve ove informacije odjednom, pa je u nekim situacijama prikladnija. Stat je, zapravo, prilagodba sistemskog poziva stat(), ali i funkcije koja se pojavljuje u većini programskih jezika.

Pogledajmo ispis naredbe stat, na istoj datoteci koju smo rabili u članku o naredbi touch:

Vidimo mnoštvo informacija, kao što je veličina (ovdje je 0, kao i broj blokova koji zauzima datoteka). Naredba prepoznaje veličinu 0, pa ispisuje "regular empty file", inače bi riječ "empty" bila izostavljena.

Nadalje, ispisuju nam se podaci o uređaju u heksadecimalnom obliku, te inode broj. Inode je broj jedinstven za svaku datoteku i direktorij na datotečnom sustavu. Vrijednost "Links" označava koliko hard linkova na tu datoteku postoji, a u ovom slučaju to je samo jedan - njen vlastiti. Simbolički linkovi se ne ubrajaju u ovu vrijednost.

Sljedeće što ćemo saznati su prava pristupa, u oktalnom i simboličkom obliku, kao i tko je vlasnik i grupa datoteke (opet, u numeričkom i simboličkom obliku).

Zadnje što ćemo saznati su vremena pristupa ("Access"), promjene ("Modify") i promjene inodea ("Change"). Drugim riječima, ATIME, CTIME i MTIME. Ovo zaslužuje malo pojašnjenja.

ATIME, ili "Access Time", je vrijeme zadnjeg pristupa datoteci. Ovo uključuje uređivanje i pregledavanje preko naredbi **cat**, **less** i sličnih, te promjenu atributa datoteke. Vrijeme zadnjeg pristupa možete vidjeti i preko naredbe ls, u obliku "**ls -lu**". Osvježavanje zadnjeg pristupa datoteci ponešto smanjuje performanse, a kako taj podatak obično nema neku vrijednost, sistem administratori znaju isključiti generiranje ovog zapisa kako bi dobili na performansama datotečnog sustava.

CTIME, ili "Change Time", nikako ne treba brkati s pojmom "Creation Time". Ovaj podatak na Unix/Linux sustavima zapravo ne možemo doznati. CTIME se odnosi na vrijeme promjene podataka u datoteci, ali i promjene na inode broju (ovo uključuje promjenu vlasnika ili prava pristupa preko chown/chmod naredbi). Vrijeme zadnje promjene možete vidjeti i preko naredbe ls, u obliku "**ls -lc**".

MTIME, ili "Modification Time", se mijenja kad se mijenja sadržaj datoteke. Ovo vrijeme je podrazumijevano kada rabite naredbu ls u proširenom obliku, "**ls -l**". Ovo je vjerojatno vrijednost koja vam je najzanimljivija.

Kako je preko primjera najlakše shvatiti sve gore izrečeno, navest ćemo tipične primjere iz prakse.

Vidimo da se promijenilo vrijeme zadnjeg pristupa datoteci (ATIME).

Ukoliko promjenimo prava pristupa, što automatski znači i promjenu pripadajućeg inodea, promijenili smo CTIME, ne i MTIME.

Promijenimo sada sadržaj same datoteke:

Možemo vidjeti da se promijenilo i vrijeme CTIME i MTIME.

Stat prepoznaje i druge tipove datoteka, kao što su direktoriji, *socketi*, linkovi i tako dalje, te će tu informaciju i ispisati:

Vidimo da je stat ispravno prepoznao datoteku /dev/null kao "character special file".

Moguće je dobiti informacije o datotečnom sustavu, umjesto o pojedinoj datoteci. Za to ćemo upotrijebiti opciju "-f":

Na kraju, ispis naredbe stat je moguće prilagoditi svojim potrebama, što je zgodno za pisanje skripti.

Primjer:

Naredbi stat smo rekli da ispiše samo prava pristupa datoteci, i to prvo u oktalnom zapisu, a potom i simboličkom. Što možete sve koristiti kao format pročitajte u man stranici, jer je opcija zaista mnogo.

Napomena: u vašoj ljusci (*shellu*) može postojati ugrađena naredba stat, pa provjerite o kojoj naredbi se točno radi (**zsh** ljuska ima svoju inačicu u vidu dodatnog modula).

Stat je dio paketa coreutils.

Naredba **mktemp** nam može pomoći najviše kad se rabi u skriptama koje sistem inženjeri pišu kako bi automatizirali održavanje sustava i tako si olakšali posao. Naredba **mktemp** nema puno opcija, a zadatak joj je jednostavan, ali vrlo važan: kreirati privremenu datoteku, koju će teško biti predvidjeti. Zašto je ovo bitno?

Zamislimo sljedeću situaciju. Vaša skripta mora kreirati privremenu datoteku u /tmp direktoriju, a izvršava se (primjerice) iz *crona* i pod ovlastima *root* korisnika. Ime datoteke je /tmp/poruka.txt.

Znajući ove činjenice, napadač (koji nema nikakvih povišenih privilegija!) može napraviti ovo:

Dakle, napravio je simbolički link "poruka.txt", koji pokazuje na ključnu datoteku sustava, i u koju ne može ništa pisati niti brisati. Provjerimo:

Dakle, korisnik nema pravo pisanja u datoteku. Sada, kada se vaša skripta pokrene (pod ovlastima root korisnika), događaju se *zanimljive* stvari. Simulirat ćemo pokretanje skripte i sami zapisati nešto u datoteku /tmp/poruka.txt.

Dakle, uz određene preduvjete, bilo koji korisnik može efikasno omesti rad poslužitelja, a i počiniti znatnu štetu! Nemojte se tješiti dvojbenom činjenicom da "korisnik ne zna točno ime privremene datoteke" ili da "svoje skripte ne pokrećete kao root korisnik", nego jednostavno izbjegnite ovakvo nesigurno kreiranje datoteka.

Ovdje će vam pomoći naredba **mktemp**. Ona se najčešće pokreće ovako:

(pokrenuta je dvaput da se može vidjeti da je ime svaki put drugačije)

Naredba prima predložak po kojemu će generirati ime, što označavamo sa velikim slovom "X". Taj dio imena će biti slučajni dio, ali možete navesti i fiksni dio ("tmp-"). Mktemp će tom predlošku kreirati praznu datoteku, te ispisati njeno ime. Kako postoji (mala) šansa da već postoji takva datoteka, najčešća uporaba naredbe mktemp u skriptama je:

Dakle, u varijablu $TMPFILE stavljamo ime novostvorene datoteke, te u nju pomoću naredbe "echo" upisujemo koji god tekst želimo. S privremenom datotekom dalje možemo raditi što god nam je volja, ali nemojte je zaboraviti obrisati na kraju (želimo čist sustav, zar ne?).

Ukoliko vas zanima što predstavlja "|| exit 1", objasnit ćemo i to. U slučaju da iz nekog razloga naredba mktemp ne uspije kreirati datoteku, skripta će završiti izvršavanje uz izlazni kod (*exit code*) "1".

Na Unix i sličnim sustavima izlazni kôd uspješno obavljene operacije je uvijek "0", dok svi izlazni kodovi različiti od nule znače neuspjeh.

Nešto više o naredbi mktemp možete pročitati u njenoj man stranici, iako naredba ne podržava previše opcija.

Zaporke su i u ovo vrijeme biometrijskih načina autentikacije nezamjenjive u bilo kojem računalnom sustavu. Iz ovog razloga nužno je odabrati za sebe i svoje korisnike dovoljno sigurne, ali i pamtljive zaporke. Nažalost, ukoliko korisnicima prepustimo da sami odabiru zaporke, to će najčešće biti imena ljubimaca, bliže rodbine ili nešto slično, što je podložno napadu socijalnim inženjeringom. Također, često će rabiti pojmove koji se nalaze u raznim rječnicima, primjerice "firefly1" ili slično, što bilo koji program za razbijanje zaporki pronalazi u nekoliko sekundi.

S druge strane, ukoliko korisnicima definirate preteške zaporke, oni će ih negdje zapisati - tipčan

slučaj je post-it listić zalijepljen na monitoru ili ispod tipkovnice. Dakle, što nam preostaje? Pa, možemo probati generirati sigurne zaporke, koje će korisnici na neki način ipak moći zapamtiti. Najčešći alat za to je naredba **pwgen**.

Pwgen će generirati zaporke od 8 znakova, ali manje-više po slogovima, tako da ih je moguće izgovoriti. Točnije, mogu ih lakše izgovoriti korisnici s engleskog govornog područja, nego korisnici s drugih govornih područja. No, i ovako ćete pokretanjem naredbe dobiti popunjen cijeli ekran, pa možete odabrati zaporku koja je i nama najlakša za zapamtiti:

Recimo, kandidati za dobru zaporku bi mogli biti nizovi "Seeje2se", "poMe0she" ili "Omae7usa". Ukoliko ne nađete ništa lako za zapamtiti, jednostavno pokrenite naredbu ponovo i novi ekran s mnoštvom drugih zanimljivih zaporki je tu.

Način na koji naredba pwgen radi je moguće promijenti s nekoliko opcija. Navest ćemo, kao i uvijek, najzanimljivije.

Ukoliko želimo jako sigurne zaporke, možemo upotrijebiti opciju "**--secure**":

Opcija će generirati u potpunosti slučajne zaporke, pa možete očekivati da će ih korisnici negdje zapisati. Ova opcija je stoga najkorisnija kada generirate zaporke za root korisnika ili korisnika pod kojim će se neki servis vrtiti pa nije bitna kompleksnost zaporke (dapače, poželjna je što sigurnija zaporka). Za još sigurnije zaporke dodajte i opciju "**--symbols**", koja će ispisati barem po jedan posebni znak u zaporci:

U dijametralno suprotnoj situaciji, moguće je generirati nesigurnije, ali pamtljivije zaporke s opcijama "**--no-capitalize**" i "**--no-numerals**":

Opcija "--no-capitalize" isključuje ispis velikih slova, dok opcija "--no-numerals" isključuje ispis brojki. Tako dobivene zaporke je moguće, teoretski, lakše zapamtiti, no to ovisi o pojedincu.

Kada naredbu želimo rabiti u skriptama, ili jednostavno želimo samo ispis samo jedne zaporke po stupcu, rabit ćemo opciju "**-1**" i "**--num-passwords=<broj>**" (ili kraće "**-N**"):

Ukoliko želimo zaista samo jednu zaporku, dovoljno je upisati:

Naredba pwgen nije jedini način za dobiti sigurne zaporke, a jedan od načina je rabiti naš on-line generator čitljivih zaporki, koji pokušava prilagoditi zaporke našem jeziku. No, nije ga moguće (na lak i pouzdan način) rabiti u skriptama, pa je dobro poznavati što pwgen može.

Sve ostale opcije i upute možete naći u standardnoj man stranici naredbe pwgen.

Pa dobro, možda naredba iz naslova ovog članka i nije baš toliko nepoznata, ali jedna od njenih funkcionalnosti je ostala prikrivena za priličan broj korisnika. Na Debianu ova naredba dolazi u pratnji skripte lesspipe koja je u stvari glavna tema ovog članka.

Linux posjeduje priličan broj naredbi s kojima možete ispisati sadržaj datoteke. Naredbe cat i more su standardne naredbe u svim verzija linuxa i korisnici se u počecima rada u shellu naviknu na njih. I tako, po nekakvoj inerciji, korisnici uobičajeno za čitanje tekstualnih datoteka ostanu na korištenju naredbe more.

Less

"Less is more" je fraza koja osim u minimalizmu vrijedi i kod naredbe less. Nju na Debianu morate instalirati kao zaseban paket (apt-get install less). Korisnik vrlo brzo uvidi prednosti naredbe less. Osim što možete strelicama scrollati gore-dolje po datoteci koju čitate, možete pretraživati datoteku prema naprijed i prema nazad ili pak koristiti kratice poput SHIFT+g za doći do dna velike datoteke.

Primjer pretrage prema naprijed:

Pozicioniranje na 534 redak

Što s .gz i .bz i ostalim netekstualnim datotekama?

Kako sysadmini uobičajeno pretražuju po velikim datotekama poput syslog ili mail.log, a pri rotaciji logova takve datoteke bivaju gzipirane, osim naredbe cat i more, korisnici brzo upoznaju naredbe zcat i zmore koje sa sobom donosi paket gzip. Naravno, da bi sve bilo na svom mjestu postoji i naredba zless koju možete koristiti na jednak način za čitanje gzipiranih datoteka.

Ali što ako biste u shellu željeli pročitati sadržaj primjerice pdf datoteke. Vrlo brzo biste primjetili da vam ni jedna od dosada navedenih naredbi ne pomaže. Osim ako u postupak ne uključite lesspipe, koji na debianu dolazi s naredbom less. Lesspipe i lessfile su ulazi preprocesori za naredbu less. U originalnoj verziji se ponašaju ponešto različito, ali na debianu je lessfile u stvari linkan na lesspipe.

Opciju lesspipe možete uključiti na više načina. Najjednostavniji je da u shellu unesete naredbu:

ili da tu liniju jednostavno dodate u ~/.bashrc kako bi bila izvršena pri loginu.

Za početak, pokušajte sada naredbom less pročitati neku od gzipiranih datoteka, primjerice:

Ukoliko imate instaliran neki od pdftotext konvertera, primjerice paket poppler-utils za squeeze ili paket xpdf-utils za lenny, primjetit ćete da

na ekran izlazi u prilično čitljivom formatu.

Slično vrijedi za .doc datoteke za čije čitanje morate imati instaliran catdoc paket.

Osim što ovom naredbom možete izlistati .iso, .rar, .tar.gz i gomilu drugih datoteka, možete pokušati u shellu izvršiti sljedeću naredbu:

Naravno, ne možete očelivati da u shellu vidite sliku, ali morate priznati da je i ovdje ispis less naredbe prilično impresivan.

Naredba "xargs" je jedna od onih za koju nikad niste čuli i nikad je niste rabili, ili je rabite stalno i to bez razmišljanja o dodatnim opcijama i mogućnostima. Ukratko, naredba xargs izvršava izlaz druge naredbe na način na koji vam to trenutno odgovara. Sličnu funkciju ima opcija "-exec" naredbe find, ali xargs je daleko fleksibilniji i jednostavniji za uporabu od često predugačkih konstrukcija naredbe find.

Naredbu **xargs** možete rabiti uz bilo koju naredbu koja ispisuje nešto na standardni izlaz, no najčešće se rabi baš uz naredbu find.

O naredbi **find** smo već napisali članak na adresi /sys-trek-objave/osnovni-alati-za-administraciju-linuxa/#naredba-find, a sve što smo tamo opisali možete povezati s xargs i učiniti s tim datotekama što vam u tom trenutku treba (kopirati ih, prebaciti, obrisati...).

Osnovna i najčešća uporaba je neka jednostavna operacija nad nekim skupom datoteka:

ili

Ovakva će kombinacija naredbi obrisati sve slike iz tekućeg direktorija. Zašto nismo upotrijebili jednostavno "rm *.jpg"? Ukoliko slika ima jako puno (tisuće), može se pojaviti greška "*Argument list too long*". Iako je ova greška danas rijetka, ipak se može pojaviti, a problem rješava kombinacija naredbi find i xargs.

I napomena, bez opcije "-maxdepth 1", find bi našao, a xargs s naredbom rm rekurzivno obrisao sve slike u direktorijima ispod trenutnog. Dakle, pripazite kod uporabe da ne biste obrisali nešto što niste htjeli. Iz tog razloga preporučujemo da rabite "| xargs echo rm" prije izvršavanja prave naredbe. Tako ćete moći vidjeti što će se dogoditi, i na vrijeme reagirati ukoliko rezultat nije ono što želite. Naredbu echo nakon zadovoljavajućeg rezultata jednostavno obrišite.

Još jedan čest slučaj uporabe naredbe xargs uključuje operacije nad datotekama koje imaju razmake ili specijalne karaktere u imenu. Rezultat bez dodatnih opcija neće biti ono što ste očekivali, jer će imena datoteka biti razlomljena na praznim mjestima:

Vidimo da je datoteka datoteka.txt uredno obrisana, ali datoteka s razmacima u imenu nije. Da bi to izbjegli, upotrijebite sljedeće opcije:

Opcije "-print0" i "-0" označavaju da se parametri ne lome na praznim mjestima ("whitespace", što je skupni naziv za space, tab i *newline* karaktere), nego se nazivi terminiraju karakterom NULL (ASCII kod 0, u programskim jezicima se označava sa "\0").

Opcija "-print0" naredbe find čini upravo to, i umjesto da svoj ispis terminira sa znakom novog retka, terminira ga sa znakom \0, što je upravo ono što xargs očekuje uz opciju "-0".

Povezano s ovim, pretpostavljeno je ponašanje naredbe xargs da argumente ispisuje u jednom retku:

Jasno da ovakva naredbena linija ne pomaže puno, pa ćemo upotrijebiti opciju "-n". Ona ograničava ispis argumenata na određeni broj redaka, a mi ćemo upotrijebiti samo jedan redak, jer nam u ovom trenutku tako odgovara:

(Kraći i ispravniji zapis je zapravo xargs -n1 ping -c1 < datoteka.txt, rezultat je jednak).

Do sada smo argumente stavljali na kraj naredbenog retka, ali kako ćemo ubaciti argumente ispred nekog drugog argumenta? Tome služi opcija "-I" (koja automatski uključuje i opciju "-n1").

Opciju "-I" ćemo rabiti primjerice kod prebacivanja ili kopiranja svih datoteka u neki direktorij:

Opcija "-I" zapravo samo određuje koji će niz znakova biti "varijabla" gdje će se smještati ime trenutnog argumenta, no uobičajeno je za to rabiti "{}". Dodatno, ukoliko želimo da se na ekran ispisuje što se trenutno izvršava, upotrijebite opciju "-t":

U datoteci datoteka.txt se nalazi popis datoteka koje treba prebaciti, no isto smo mogli postići naredbama ls, find, ili bilo kojim drugim koje generiraju nekakav popis. Na sličan način možete kopirati datoteke ili ih preimenovati.

Xargs se može rabiti i s interaktivnim naredbama. Ukoliko u datoteci popis.txt imate popis datoteka koje treba izmijeniti, omiljeni editor (vim, joe, pico...) možete pokrenuti i ovako:

Što se tiče sadržaja datoteka, ni ovdje nema prepreka:

Gornjom kombinacijom naredbi pretražili smo cijeli /etc direktorij u potrazi za ključnom riječi "address".

Vjerujemo da smo opisali najzanimljivije načine uporabe naredbe xargs, a za dodatne primjere jednostavno pretražite web.

Za naredbama **pwck** i **grpck** nećete toliko često posezati, ukoliko je sa sustavom sve u redu, te dodajete i brišete korisnike uredno i na konzistentan način. No, s vremenom se mogu pojaviti "fantomski" korisnici ili neki drugi problemi, koje ste nehotice prouzrokovali vi, ili je do toga došlo zbog nadogradnje sustava ili povrata podataka sa starijeg backupa. Najčešće, problemi nastaju nakon ručnog prebacivanja podataka o korisnicima s jednog poslužitelja na drugi, kad je grešku najlakše napraviti.

Naredba pwck istovremeno provjerava datoteke /etc/passwd i /etc/shadow, a u njima provjerava broj polja (kojih mora biti 7), numeričke oznake korisnika i grupa, postoje li radni direktorij i korisnička ljuska (shell). Također, provjerava se je li ime korisnika jedinstveno na sustavu, a dupli unosi će biti ponuđeni za brisanje.

Ukoliko pokrenete naredbu pwck bez ikakvih argumenata, dobit ćemo otprilike ovakav ispis:

Dakle, pwck je prijavio određene probleme s HOME direktorijima, grupama i korisnickim ljuskama. Neke probleme možemo zanemariti, poput radnih direktorija sistemskih korisničkih računa, jer su neki i zamišljeni da nemaju radni direktorij (poput korisnika "nobody"). No, korisnik "darko" bi morao biti u nekoj grupi koja je ispravno definirana na sustavu, pa ćemo je dodati:

Dakle, korisnik "darko" je u grupi koja nije uvedena u /etc/group, što smo provjerili s naredbom grep. S naredbom groupadd dodali smo novu grupu "darko" i forsirali numeričku oznaku (GID) 1000. Nakon toga su sve datoteke korisnika darko "automagično" dobile i ime grupe. No, u vašem slučaju se možda radi o korisniku kojeg treba obrisati, pa postupite kako svaki pojedinačni slučaj zahtjeva.

Dakle, naredba pwck će pomoći u sređivanju datoteka /etc/passwd i /etc/shadow. Ukoliko ste sigurni da je datoteka /etc/passwd u potpunosti ispravna možete upotrijebiti naredbu **pwconv**, koja će sinkronizirati te dvije datoteke. Ali, s njom oprez, jer će obrisati svakog nepostojećeg korisnika u datoteci /etc/shadow , što u vašem slučaju može biti na stotine korisnika koji će ostati bez zaporke. Bolje je prije toga maksimalno raščistiti situaciju s naredbom pwck i grpck, a tako uostalom savjetuju i u pripadajućoj man stranici.

Nakon što smo popravili datoteke sa zaporkama, možemo pogledati ima li kakvih problema s datotekama koje čuvaju podatke o grupama, za što ćemo upotrijebiti ćemo naredbu **grpck**:

Datoteka /etc/gshadow bi trebala sadržavati enkriptirane zaporke za grupe, no to se rijetko rabi. U ovom slučaju možete odgovoriti potvrdno na sva pitanja, što će dodati grupe iz /etc/group u /etc/gshadow. No, neće sadržavati nikakve zaporke ukoliko ih ni prije niste rabili.

I u /etc/group mogu zaostati obrisani korisnici:

U ovom slučaju, u grupi adm je zaostao obrisani korisnik "petar", pa taj unos možete obrisati (ili ponovo dodati korisnika, ukoliko je greškom bio obrisan). Mogu vam se javiti i ovakve poruke:

Sustav vam na ovaj način poručuje da /etc/group i /etc/gshadow nisu sinkronizirane, a najlakše rješenje je upotrijebiti naredbu **grpconv**:

Sustav više nema "primjedbi", što se može vidjeti iz izlaza naredbe. Naredba grpconv pretvara sve unose iz /etc/group u sintaktički pravilne unose za /etc/gshadow. Iz ovoga proizlazi da, isto kao u slučaju /etc/passwd, treba paziti na ispravnost datoteke /etc/group.

Naredbe pwck i grpck imaju opciju "-r" (read-only), koja neće napraviti nikakve promjene na sustavu, ali će pokazati što ne valja. Također, obje imaju opciju "-s" s kojom postižemo sortiranje unosa u ovim datotekama po UID-u, odnosno GID-u.

Radi potpunosti, spomenut ćemo srodne naredbe **pwunconv** i **grpunconv**, koje će spojiti zaporke natrag iz /etc/shadow u /etc/passwd i iz /etc/gshadow u /etc/group. No, ne vjerujemo da ćete ovo ikada trebati napraviti.

Na kraju, ponovit ćemo tvrdnju s početka, ove dvije, odnosno četiri naredbe nećete često rabiti, ali u slučaju potrebe pomoći će vam da sustav osposobite u što kraćem vremenu.

Za dohvat podataka o korisnicima možemo, pored ostalih, rabiti naredbu **getent**. Ova naredba može dohvatiti i druge podatke, jer može čitati iz drugih "baza" na sustavu, poput baze hostova (/etc/hosts), servisa (/etc/services) ili grupa korisnika (/etc/group). Primjerice, ukoliko želite saznati informacije o korisniku iz baze /etc/passwd, pokrenut ćemo getent na ovaj način:

Grupe korisnika možemo saznati ovako:

Ukoliko vas ovaj ispis podsjeća na običan ispis koji možete dobiti naredbom **grep**, u pravu ste, ali samo djelomično. Naime, getent vadi podatke iz baza onako kako ih **sustav** vidi (konfiguracija se nalazi u datoteci /etc/nsswitch.conf). Ovo konkretno znači da ćete dobiti podatke i iz LDAP baze, ukoliko ste na taj način konfigurirali sustav. Slično je i za starije sustave, bazirane na NIS-u.

Ukoliko želimo ispisati cijelu bazu, jednostavno navedite bazu bez dodatnog parametra:

U gornjem primjeru smo odmah demonstrirali kako dobiti podatke iz treće baze, popisa hostova koje ste unijeli u datoteku /etc/hosts.

Sljedeća baza je baza servisa, pa ukoliko želimo saznati na kojem portu sluša LDAP servis i njegova enkriptirana inačica, otkucat ćemo:

Ostale dvije baze su networks i protocols. Baza protokola nam nije previše zanimljiva, jer sadržava samo broj i alias protokola unutar TCP/IP zaglavlja, te je tako možda više zanimljivija programerima. Baza mreža sadržava bazu podataka o mrežama, koje se nalaze u datoteci /etc/networks, koju rabe naredbe netstat i route.

Zaključak je da je naredbu getent najbolje rabiti unutar skripti umjesto naredbe grep, jer je "vjerodostojnija" (neće ispisati ništa ukoliko korisnik ne postoji na sustavu). Također, sigurnije ju je rabiti nego grep, jer uvijek vraća jedan rezultat, ili ga uopće ne vraća.

Ukoliko, u nekoj brzini, idemo rabiti naredbu grep, možemo dobiti više rezultata, primjerice "grep -i ivic /etc/passwd" će ispisati podatke korisnika "ivica", ali i bilo kojeg drugog korisnika kojem bilo koji podatak iz GECOS polja (primjerice prezime "Sljivic"), završava na "-ivic". Ovo je pogotovo nezgodno ukoliko getnet rabite u skripti, jer na jednom sustavu može raditi sasvim u redu, a na drugom (gdje ima više korisnika), neće raditi kako treba, što može napraviti i "štetu" na sustavu.

Linux, a ranije i razni Unix sustavi, u osnovnoj instalaciji obično imaju sve alate za nadzor koji trebaju sistem inženjerima koji taj sustav održavaju. No, s vremenom su neki alati prerasli u preglomazne programe, s gomilom opcija i kriptičnim ispisom. Neki su davno sazreli za promjene, ali toliko drugih stvari ovisi o njima, odnosno o njihovom ispisu, da bi se stvorilo daleko više problema nego bi ih se riješilo. Zato je nekada jednostavnije napisati novi alat koji radi samo jednu stvar, ali zato to radi dobro. Mislimo da je dobar primjer alat **nethogs**.

Nethogs, kako mu i ime kaže, prati mrežni promet po procesima i ispisuje one koji trenutno najviše "troše". Ispis je vrlo sličan ispisu programa **top**, pa razdoblja privikavanja nema:

U interaktivnom načinu rada program prima samo dvije opcije: "m" za promjenu jedinica iz bajtova u kilobajte i megabajte, te trenutnu brzinu koju proces postiže u mrežnom prometu:

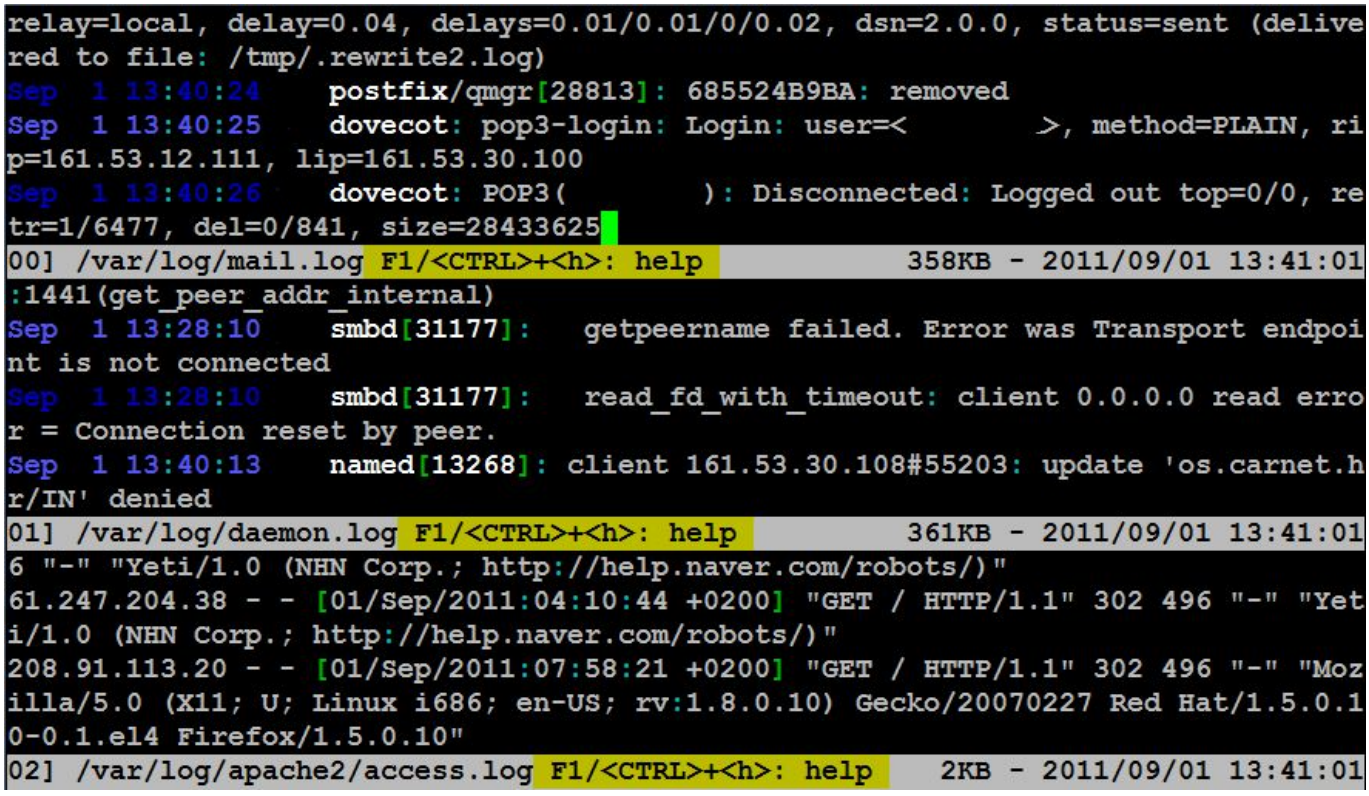
Druga opcija je "q", koja jednostavno služi za izlaz iz programa.

Ostale (malobrojne) opcije možete saznati preko opcije -h:

Da logove na poslužitelju treba redovito pregledavati, zna svaki sistem-inženjer. Jednostavno, bez logova bi posao sistem-inženjera bio nemoguć. Iako imamo automatizirane programe (tipa logwatch, fail2ban ili OSSEC) koji nam pomažu u detekciji potencijalnih problema (poglavito sigurnosnih), neophodno je ponekad logove pogledati "uživo". Ovdje se možemo poslužiti programima tail i less, koji mogu pratiti logove onako kako se *pune*, ali imamo odličan alat za praćenje više logova odjednom - **multitail**.

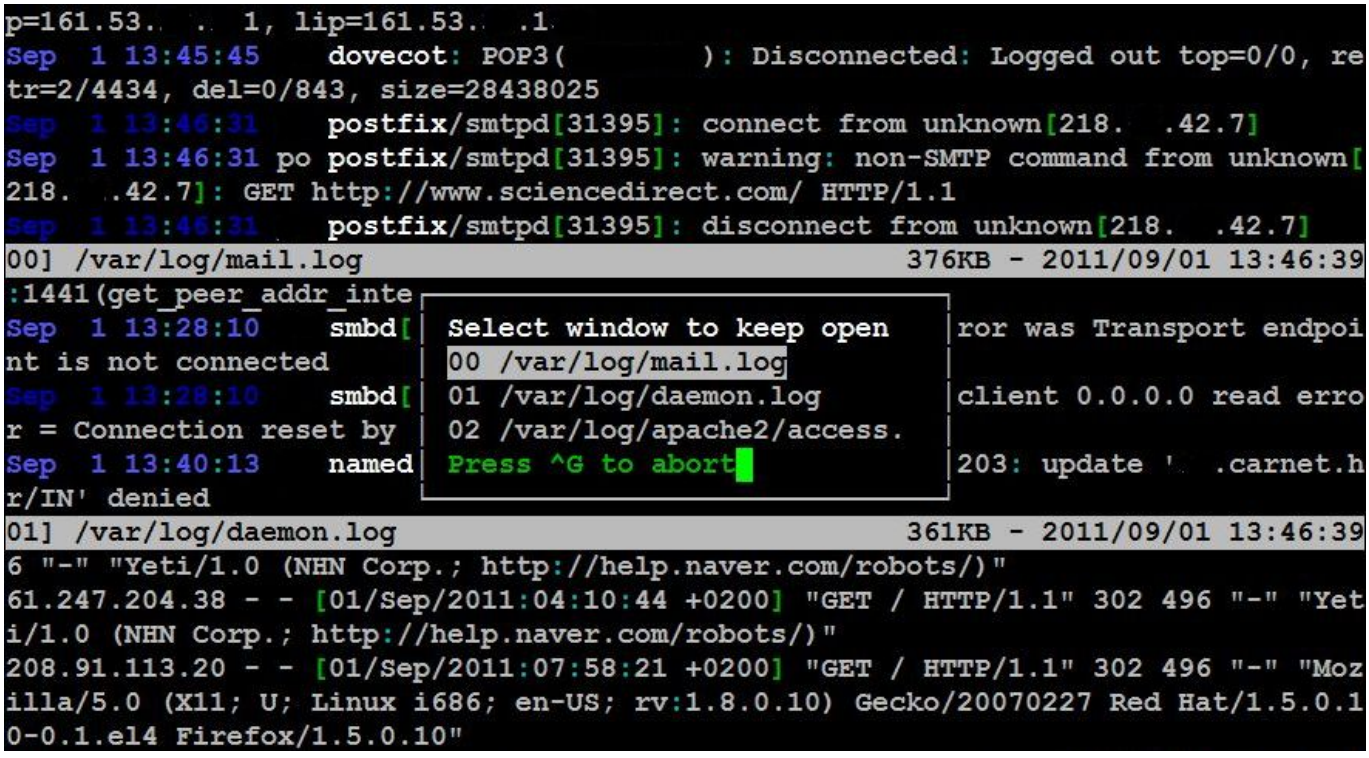
Kao i programčić "nethogs", kojeg smo opisali u prethodnom članku u ovoj seriji, tako i multitail slijedi jednostavnu logiku: raditi jednu stvar dobro, i ne zbunjivati korisnika bespotrebnim opcijama koje u većini slučajeva neće nikada rabiti. Pokretanje programa je standardno, samo treba nakon imena naredbe treba navesti logove koje želimo pratiti:

Multitail radi preko vrlo poznatog *ncurses* sučelja za tekstualne terminale, što znači da ima sustav prozora koji pokušava postići dio funkcionalnosti klasičnih grafičkih GUI-ja. Nakon pokretanja, bit će prikazana tri prozora kao na slici:

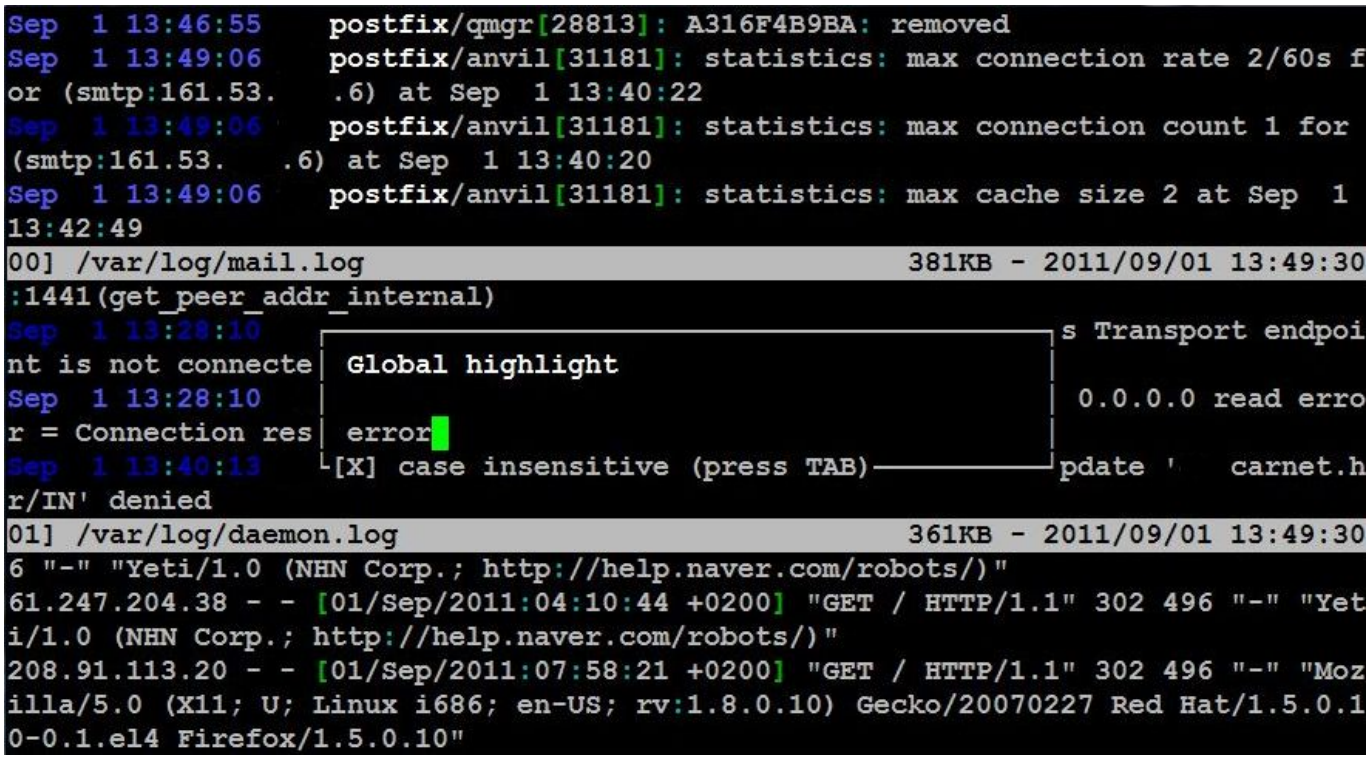


U svakom od tri prozora bit će prikazano nekoliko zadnjih redova odabranih log datoteka. Multitail pokušava biti uslužan, pa može obojati određene dijelove logova drugom bojom, primjerice datum je označen plavom bojom, naizmjenično svjetlijom i tamnijom nijansom. U svakom trenutku možete dobiti pomoć ukoliko pritisnete kombinaciju tipki **<CTRL+h>**. U prozoru pomoći možete pregledati sve dostupne naredbe sa standardnim **Page Up** i **Page Down** tipkama, a prozor gasite sa **<CTRL+g>**.

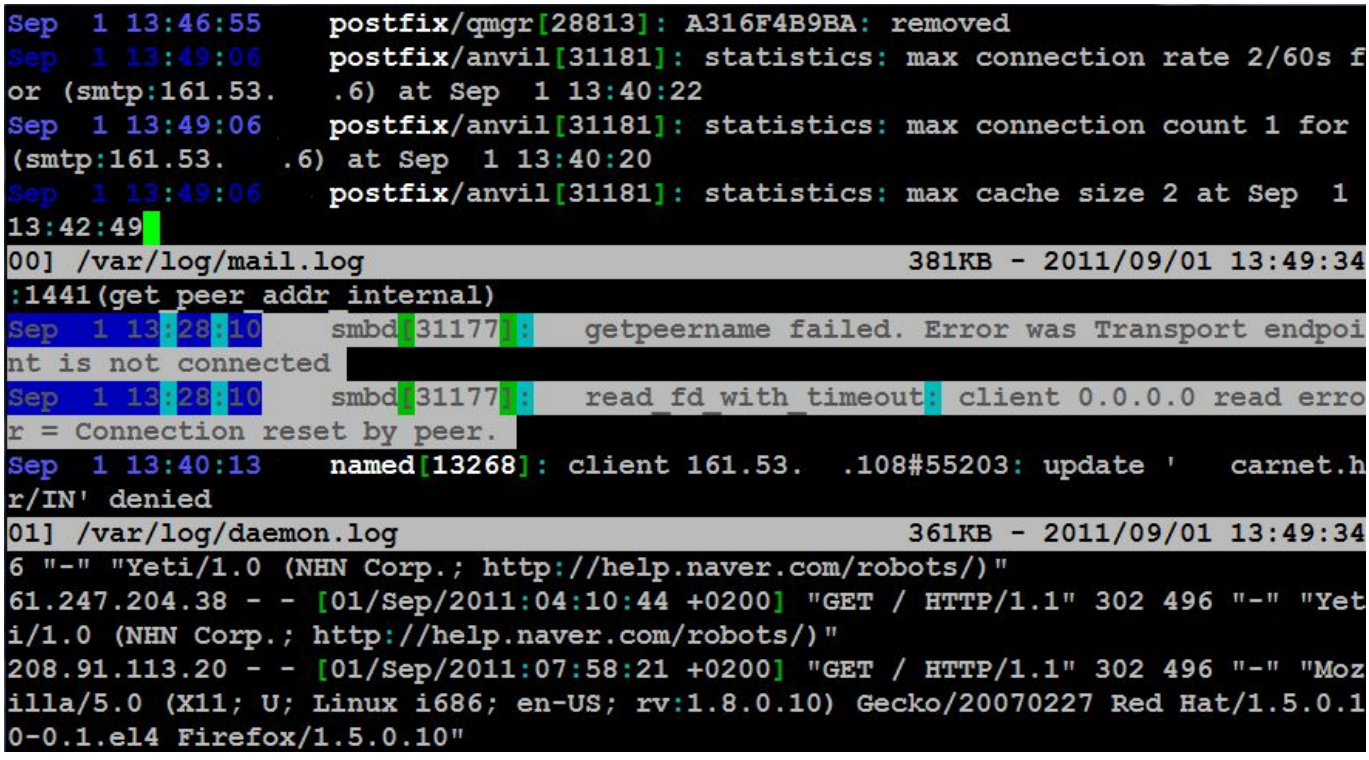
Ukoliko u nekom logu vidite nešto vrijedno pažnje, možete privremeno ugasiti sve prozore osim odabranog. Ovo možete napraviti pomoću tipke **"u"**, nakon čega odabirete željeni prozor:



Nakon što završite, možete se vratiti u prethodni prikaz svih prozora s tipkom **"U"** (naravno, ovdje je riječ o kombinaciji **<Shift+u>**). Vizualni pregled očima nije uvijek i najbrži, zato multitail posjeduje funkciju traženja, koju možete pozvati s **"/".** Slično ovoj**,** sa kombinacijom **<Shift+/>**, možete dobiti **highlite** funkciju [1], kao na slici:



Mi smo potražili najzanimljiviju riječ koju možemo naći u logovima: **error**. Nakon što multitail pronađe traženu riječ, označit će cijeli redak u kojemu je pronašao traženu riječ:



Označavanje redaka ostaje i ako prijeđemo u način rada s jednim prozorom.

Moramo priznati da smo vas malo prevarili, jer multitail nema samo dvije-tri, nego preko dvadeset

opcija. Tako, multitail može pratiti izlaz naredbi (STDOUT), može izvršavati proizvoljne naredbe ukoliko se pojavi određeni string (točnije *regex*) unutar log datoteke ili u izlazu naredbe, a očekivano podržava prilagodbu boja i sučelja. Smatramo da su za ove naprednije stvari primjereniji već spominjani fail2ban, a pogotovo OSSEC, pa nećemo ulaziti dublje u ove mogućnosti multitaila.

[1] na tipkovnici s HR rasporedom tipaka znak **/** se dobija s kombinacijom **<Shift+7>**, dakle već rabimo Shift. Kako onda dobiti kombinaciju tipaka **<Shift+/>**? Postoje (barem) dva načina... odgovorite u komentarima.