Operacijski sistem Unix

Uvod - Zagonska veriga - Delovanje in zaustavitev - Datotečni sistem - Virtualne konzole - Delovanje v ospredju in ozadju - Zgradba in pisanje ukaza - Tok podatkov in cevovodi - Specialni lupinski znaki - Ukazni vzdevki in spremenljivke - Datotečna meta imena - Tekstovni meta nizi

Uvod

Unix je večopravilni in večuporabniški operacijski sistem. K. Thompson in D. Ritchie sta ga napisala 1974 v programskem jeziku C; vanj sta vložila 10 inženirskih let dela. Unix je bil napisan za računalnik PDP-11, kasneje pa nameščen na druge stroje. Brezplačno okleščeno kopijo Unixa, Linux, je 1991 za PC napisal L. Torvalds. Od takrat sta se Linux in programska oprema zanj razvila in razširila po vsem svetu. Unix/Linux je najboljši operacijski sistem, kar jih je do sedaj naredilo človeštvo. Je majhen, preprost, modularen, robusten, hiter in fleksibilen. Zlasti je primeren za tehnike, inženirje in znanstvenike.

Zagonska veriga

BIOS. Ko prižgemo računalnik, začne procesor izvrševati program BIOS (Basic Input Output System), ki je zapečen v posebnem pomnilniškem čipu ROM. BIOS zna dostopati do priključenih enot: magnetnega diska, optičnega diska, polprevodniškega vtiča in drugih. BIOS dostopa do teh enot po določenem zaporedju; prva enota, kjer najde operacijski sistem, obvelja. Preden procesor začne delati po BIOS, malo počaka. Če takrat pritisnemo posebno tipko, navadno [F2], lahko v BIOS vnesemo vrstni red dostopanja do enot, zaščito z geslom in še marsikaj drugega.

LILO. Naj bo operacijski sistem na trdem disku. Na trdem disku pogleda BIOS v prvi sektor, MBR (Master Boot Record). Tam je nameščen program LILO (Linux Loader), ki se začne izvajati. LILO ima v sebi zapisano, kje na disku je nameščeno jedro (kernel) operacijskega sistema Linux. Naloži ga v pomnilnik in ga začne izvajati. Preden LILO naloži jedro, izpiše njegovo ime, recimo linux, in obvestilo "boot:" ter malo počaka. Sedaj lahko odtipkamo kakšne parametre in LILO jih bo posredoval jedru.

Kernel. Jedro ima v sebi zapisano, kje na disku je datotečni

sistem Linux; tam so zapisani različni programi. Iz datotečnega sistema se naloži v pomnilnik izvršni program init in se začne izvajati.

Init. Init pogleda v datoteko inittab. Tam so zapisana imena zagonskih skriptov, ki se naj izvršijo. Tipično sta to najprej rc.S in nato rc.M. Zapisano je tudi, kaj se zgodi s sistemom, če na konzoli pritisnemo [Ctrl-Alt-Del]; tipično se sistem izključi.

Zagonski skripti. Izvrši se vse, kar je zapisano v zagonskih skriptih. Iz obeh skript se kličejo drugi skripti. Ko so vsa skripta izvršena, zažene init program agetty, ki na konzolo zapiše "login:" ter čaka, da se bo kak uporabnik priključil. Računalnik je pripravljen za delo.

Delovanje in zaustavitev

Ukazna lupina in uporabnik. Uporabnik odtipka svoje ime in geslo. Init zažene poseben program - ukazno lupino, tipično bash, ki se javi z znakom "$". Vse, kar sedaj tipkamo, prevzame bash in temu ustrezno ravna. Tipkamo ukaze. Nekatere zna izvršiti bash sam, za druge pa pokliče ustrezne programe. Nekateri programi po zagonu komunicirajo z uporabnikom. Na koncu se odjavimo s posebnim ukazom in na zaslonu se spet pokaže "login:". Preden se bash oglasi, prebere datoteko .profile, ki določa njegovo ravnanje.

Uporabnik root. Uporabnik z imenom root je priviligiran. Ko se prijavi, se mu bash oglasi z "#". Uporabnik root lahko izvršuje ukaze, ki jih drugi ne morejo. Specialno lahko v LILO na MBR zapiše zaščito z geslom ter ime in lokacijo jedra; v jedro lahko zapiše lokacijo datotečnega sistema; popravlja lahko zagonske skripte; dodaja veljavna uporabniška imena in gesla ali jih odvzema in podobno. Končno lahko odtipka tudi ukaz, ki zaključi delo operacijskega sistema Linux, nakar lahko računalnik ugasne.

Okna X. Uporabnik lahko s posebnim ukazom požene okenski strežnik X ter izbran vmesnik, recimo ion. Izbrani vmesnik je zapisan v uporabnikovi datoteki .xinitrc. Vmesnik omogoča, da se na zaslonu odpirajo in zapirajo okna, v katerih "so" programi. Odtlej uporabnik komunicira s tem vmesnikom in s programi v oknih. Večinoma komunicira tako, da z miško klika po raznih menijih. Delo konča z ukazom, ki je odvisen od uporabljenega vmesnika. S tem se vrne nazaj k bash.

Hierarhični datotečni sistem

Izvršni binarni programi ter njihovi vhodni ali izhodni podatki so zapisani v datotekah, tipično na magnetnem disku. Datoteke so osnovni sestavni del datotečnega sistema. Drugi del so imeniki; to so posebne datoteke, v katerih so zapisana imena drugih datotek. Tak imenik je torej "mapa", ki "vsebuje" druge datoteke. Imenik lahko vsebuje druge imenike. Celotna imeniška struktura ima obliko drevesa, ki raste iz enega samega osnovnega imenika.

Preko bash referenciramo datoteke z njihovim polnim imenom, ki vključuje vse zaporedne imenike od osnovnega navzdol, ločene z znakom /, ter ime datoteke same. Na primer: /usr/bin/joe. Ime imenika ali datoteke lahko vsebuje znake a-z, A-Z, 0-9, minus, plus, podčrtaj in piko. Male in velike črke se ločijo. Ime datoteke ali (pod)imenika ne sme biti daljše od 32 znakov.

Tipična struktura datotečnega sistema je naslednja:

Imenik	Vsebina
/	
/boot	Jedro
/dev	Periferne enote
/etc	Zagonski skripti
/lib	Programske knjižnice
/bin	Osnovni programi
/sbin	Osnovni sistemski programi
/usr	Dodatni programi
/etc	
/lib	
/bin	
/sbin	
/opt	Dodatni programi za X
/root	Uporabnik root
/home	
/USER	Uporabnik USER
/mnt	Priključene enote
/var	Log datoteke
/tmp	Začasne datoteke

Vsaka datoteka je sestavljena iz glave in telesa. V telesu so podatki, v glavi "atributi": kakšnega tipa je datoteka (imenik, navadna datoteka ali kaj tretjega), kdo je njen lastnik, ali je lastniku in drugim dovoljeno iz nje brati ali vanjo pisati, kdaj je bila zadnjič brana, kdaj spremenjena in še kaj. Posebni ukazi omogočajo pregled in nastavitev nekaterih atributov.

Vsak uporabnik ima svoj lastni imenik pod imenikom /home, katerega vsebino lahko z ukazi preoblikuje, kakor hoče: dodaja, briše, preimenuje in premešča podimenike in datoteke. Do drugih imenikov nasplošno nima takega dostopa, ki bi bil zanje lahko škodljiv. Uporabnik root ima neomejen dostop do vseh imenikov.

Virtualne konzole

Fizično konzolo, za katero sedimo, lahko uporabimo kot več virtualnih konzol, tipično štiri. Izbrano izmed njih dosežemo s tipko [Ctrl-Alt-F1]..[Ctrl-Alt-F4]. Na njej se pokaže napis "login:" in lahko se prijavimo. Prijavimo se lahko na več virtualnih konzol in kot različni uporabniki. Vsakokrat se nam oglasi nov bash. Potem na teh konzolah delamo in kadarkoli med njimi preklapljamo z navedenimi tipkami. Preklapljanje je možno samo v konzolnem načinu dela; v okenskem ni možno. Unix dela za vse uporabnike "hkrati"; to počne tako, da dela hipec za prvega, nato hipec za drugega in tako naprej. Hipci so tako kratki, da uporabnik ne opazi kasnitev pri svojem delu.

Delovanje v ospredju in ozadju

Bash lahko izvršuje ukaze na dva načina: v ospredju in ozadju. Ukaz COMMAND, zapisan v obliki

```
$ COMMAND ; [Enter]
```

izvrši do konca, nato šele spet pokaže svoje obvestilo; pravimo, da se ukaz izvaja v ospredju. Podpičje na koncu vrstice lahko spustimo. Ukaz, zapisan v obliki

```
$ COMMAND & [Enter]
```

pa začne izvrševati, vendar takoj spet pokaže svoje obvestilo ter s tem pripravljenost na sprejemanje novih ukazov,

medtem ko se dani ukaz še izvršuje; pravimo, da su ukaz izvaja v ozadju.

Zgradba in pisanje ukaza

Posamičen ukaz ima lahko naslednjo splošno obliko:

```
COMMAND -ABC -D NNN OPERAND1 OPERAND2 ...
```

A, B in C so enočrkovna "stikala", ki jih razume ukaz, in po njih prilagaja svoje delo, -D NNN pa je stikalo s parametrom. Posamezna stikala lahko pišemo ločeno, -A -B, ali pa skupaj, -AB. OPERAND1 in drugi so operandi, tipično imena datotek, s katerimi naj ukaz kaj počne. Stikala so specifična za vsak ukaz. Nekateri ukazi nimajo niti stikal niti operandov. Male črke so drugačni znaki od velikih.

COMMAND je lahko polno datotečno ime programa, torej vsebujoče imeniški del (na primer /bin/date), ali pa zgolj golo ime (na primer date). Če je ime golo, išče bash po vrsti po vnaprej določenih imenikih, dokler ne najde. Kateri so ti imeniki, lahko določi uporabnik s posebnim ukazom.

Pri tipkanju ukaza se lahko zmotimo. Preden odtipkamo [Enter], lahko vrstico popravljamo. Delujejo ustrezne tipke [Left], [Right], [Delete].

Nekateri ukazi so dolgi. Če natipkamo samo začetek in nato tipko [Tab], bash avtomatično dopolni ukaz do tiste dolžine, od katere naprej je več možnosti. Uporabnik nato nadaljuje s tipkanjem. Lahko pa drugič pritisne [Tab] in bash pokaže vse možnosti nadaljevanja.

Vse ukazne vrstice, ki smo jih že odtipkali, bash shranjuje. Po njih se sprehajamo s tipkama [Shift-PgUp] in [Shift-PgDown] in jih aktiviramo.

Tok podatkov in cevovodi

Kjerkoli je smiselno, so ukazi takšni, da sprejemajo vhod s tipkovnice in pišejo rezultate ter napake na zaslon. Pravimo, da čitajo iz stdin, pišejo rezultate na stdout in napake na stderr.

Če je izpis na zaslon dolg, ga ustavimo s tipko [Scroll Lock].

To pride v poštev le pri počasnem izpisovanju. Nekaj sto zadnjih zaslonskih vrstic (tistih, ki smo jih pisali mi in onih, ki jih je izpisal računalnik) je tudi shranjenih; po njih se premikamo s tipkama [Shift-PgUp] in [Shift-PgDn].

Vhod in izhod lahko preusmerimo na datoteko. Namesto na zaslon piši v datoteko OUTFILE:

```
COMMAND > OUTFILE          piši rezultate
COMMAND 2> OUTFILE         piši napake
```

Predhodna vsebina datoteke se pri tem pobriše. Če želimo v datoteko dodajati, uporabimo namesto znaka > znak >>.

Namesto s tipkovnice čitaj vhodne podatke iz datoteke INFILE:

```
COMMAND < INFILE
```

Izhod prvega ukaza lahko neposredno uvedemo v drugi progam. Pravimo, da ju povežemo s cevjo:

```
COMMAND1 | COMMAND2
```

Primer je date | wc, kar pomeni: "izpiši" datum in čas, nato pa preštej, koliko vrstic, besed in znakov je v "izpisu" ter rezulat zapiši na zaslon. Povežemo lahko tudi več ukazov.

Specialni lupinski znaki

Bash razume posebne specialne znake, ki uravnavajo njegovo delovanje, na primer izvajanje v ospredju in ozadju, preusmeritev in cevovod ter drugo. Specialnih znakov ne posreduje naprej ukazom. Če jim jih hočemo posredovati v operandih, jim moramo predhodno odvzeti specialni pomen.

Vsi specialni znaki (nekatere smo že spoznali, druge še bomo) so naslednji:

```
; &          Zaporedno in vzporedno izvajanje ukazov
( )          Grupa ukazov, izvrši v podlupini
{ }          Grupa ukazov, izvrši v tekoči lupini
< > |        Preusmeritev toka in cevovod
~ * ? [ ]    Ekspanzija datotečnih imen
$            Vsebina spremenljivke ali izhod ukaza
#            Komentar
" ' \        Odvzem pomena specialnim znakom
```

Odvzem pomena specialnim znakom:

\	Naslednji specialni znak naj bo normalen
" "	Vsak specialni znak znotraj naj bo normalen, z izjemo znaka $, ki obdrži specialni pomen.
' '	Vsak specialni znak znotraj naj bo normalen

Ukazni vzdevki in okoljske spremenljivke

Poljubnemu ukaznemu nizu znakov lahko določim njegov vzdevek. Kadarkoli potem tipkamo vzdevek, ga bash nadomesti s prvotnim nizom znakov. Tako si skrajšamo tipkanje dolgih ukazov. Vzdevek ALIAS za zaporedje tekstovnih znakov TEXT, ki vsebuje lahko tudi presledke, deklariramo takole:

```
alias ALIAS='TEXT'
```

Primer: alias date='date -ü. Bash si zapomni vzdevek, dokler smo prijavljeni. Ko se odjavimo, ga pozabi. Lahko pa vse deklaracije spravimo v datoteko .profile in bash jih prebere ter si jih zapomni vsakokrat, ko se prijavimo. Kakšne vzdevke bash trenutno pozna, ugotovimo kar z ukazom "alias". Če je alias identičen imenu kakega drugega programa, bash vedno izvrši alias, razen če mu eksplicitno odvzamemo pomen: \ALIAS.

Tekstovne nize lahko spravljamo tudi v okoljske spremenljivke. Nekateri programi namreč gledajo specifične spremenljivke in uravnavajo svoje delo glede na njihovo vrednost. Spremenljivko ENVAR deklariramo kot

```
export ENVAR='TEXT'
```

Bash jo hrani, dokler smo prijavljeni. Lahko pa jo tudi deklariramo v datoteki .profile in potem jo bash naredi vsakokrat, ko se prijavimo. Kakšne so trenutno znane spremenljivke, vidimo z ukazom "printenv".

Datotečna meta imena

Kadar želimo ukazu posredovati več imen datotek, to storimo z ustreznimi meta-oznakami ~, *, ?, [] in {} v imenu datoteke. Pri tem pomenijo oznake naslednje:

```
    ~           domači imenik
    *           poljuben niz
    ?           poljuben znak
    [abc]       eden izmed znakov a, b, c
    [a-z]       eden izmed znakov a..z
    {bin,etc}   eden izmed nizov bin, etc
```

Primeri:

```
    *, *.txt, memo.*
    fig?.gif, fig??.gif
    table[123].tex, table[1-9].tex
    memo.{txt,tex,xht}
```

Iz meta-imena naredi bash najprej celotni seznam popolnih imen, nato pa tega posreduje zunanjemu programu.

Tekstovni meta nizi

Nekateri ukazi zahtevajo kot enega izmed parametrov tekstovni "vzorec". Tekstovni vzorec (regular expression) opisuje strukturo kakšnega tekstovnega niza (vrstice). Vsebuje metaznake s posebnim pomenom:

```
    .       Katerikoli znak
    [ABC]   Znak A, B ali C
    [A-Z]   Katerikoli znak A..Z
    [^ABC]  Katerikoli znak razen A, B, C
    *       Nič-poljubnokratna ponovitev prejšnjega znaka
    ^       Začetek vrstice
    $       Konec vrstice
    +?()|   Nepotrebno
    \       Izključi specialni pomen metaznakom
```

Primeri:

```
  Vrstica, ki vsebuje vsaj en znak: .
  Celotna vrstica: .*
  Vrstica, ki se začne s #: ^#
  Prazna vrstica: ^$
  XHTML tag: <[^>]>
```