LU Linux Centrs

Linux komandrindas saskarne – konsole

Rīgā 2006. gadā.

Kas?

Komandrindas saskarne (*CLI – Command Line Interface*) ir viena no metodēm, kā sazināties ar datoru. Mūsdienās plašāk izplatītāka metode ir grafiskā lietotāja saskarne (*GUI – Graphical User Interface*).

Vienkāršākajā gadījumā komandrindas saskarne izvada uzvedni (*prompt*), ļaujot lietotājam ievadīt komandu, ko viņš apstiprina ar ievades taustiņu. Pēc tam dators komandu izpilda un atgriež izvadu teksta veidā, ja tāds ir. Vajadzības gadījumā programma var pajautāt jautājumus, uz kuriem iespējams atbildēt, ierakstot atbildi un to apstiprinot ar ievades taustiņu. Taču visu darba laiku iepriekšējās komandas un to izvads paliek redzams uz ekrāna (ja tajā ietilpst).

Sarežģītākas komandrindas programmas pēc palaišanas ieiet teksta lietotāja saskarnē (*TUI – Tekst User Interface*), pēc kā ekrāns tiek notīrīts un uz tā tiek uzzīmēts programmas logs (**teksta režīmā**), kurā iespējams darboties atkarībā no programmas tipa. TUI ir raksturīgi teksta redaktoriem, datņu pārvaldniekiem un citām programmām. Šeit uz jautājumiem rakstiski atbildēt parasti nevar, bet iespējams darboties ar dažādām saīsnēm vai pat izvēlnēm un apstiprinājumu logiem. Piemēram, teksta redaktoru *pico* un *nano* apakšā vienmēr atrodas aktuālāko saīšņu saraksts. Ja DOS esam pieraduši redzēt tādus uzrakstus kā “Ctrl+C”, “Ctrl+I”, tad UNIX-veidīgajās operētājsistēmās tie parasti attiecīgi tiek apzīmēti “^C” un “^I”. Vispopulārākā bibliotēka TUI programmu rakstīšanai UNIX-veidīgajās operētājsistēmās droši vien ir *ncurses*.

Īpaša vieta ir bibliotēkai *aalib*. Tā ļauj praktiski jebkuru GUI aplikāciju izpildīt teksta režīma vidē. Tas notiek, pārrēķinot pikseļu grupas ASCII simbolos un izvadot tos uz ekrāna. Ar šīs bibliotēkas palīdzību uz tādas spēles kā *Quake* un *Unreal Tournament* ir iespējams spēlēt 80x25 simbolu teksta režīmā. Jautājums par lietderību gan paliek neatbildēts. Ar šīs bibliotēkas palīdzību ir uzrakstīta arī populārā video atskaņotāja *xine* modifikācija – *aaxine* – kas spēj atskaņot video datnes konsolē.

Vienīgais konsoles ierobežojums, salīdzinot ar GUI, ir vizuālais ierobežojums. Mīti par skaņas, multiprocesoru, peles, USB u.c. atbalsta neesamību nav patiesi.

Kāpēc?

Pirmām kārtām – tā tas vēsturiski ir izveidojies. Daudzās vietās arī joprojām nav vajadzības pēc sarežģītas GUI saskarnes, un tas ļauj ietaupīt uz grafikas adapteriem un monitoriem. Daudzās vietās vispār nav vajadzība pēc saskarnes, un tas ļauj vispār nepieslēgt datorsistēmai ne grafikas adapteri, ne monitoru, bet to šajā rakstā sīkāk neapskatīšu.

Tātad, pēc GUI saskarnes ne vienmēr ir vajadzība. Ir gadījumi, kad CLI lietošana ir stipri ērtāka un ieteicamāka, katra situācija ir jāspēj izvērtēt, lai izvēlētos pareizo saskarni, ja vien tāda ir pieejama. Jāpiebilst, ka, lietojot GUI saskarni, praktiski vienmēr ir iespējams paralēli palaist virtuālu konsoli (CLI), taču lietojot komandrindas saskarni, nav nekādu iespēju palaist virtuālu grafisku saskarni, jo grafikas adapteris dotajā režīmā vienkārši nespēj uzzīmēt nekādu attēlu, kā vien tekstu.

Cilvēki, kas ikdienā lieto CLI saskarni, pie tās mēdz tik ļoti pierast, ka GUI izmantošana sķiet un ir lēnāka un neparocīgāka. Pētijumi ir pierādījuši, ka lietotāji, kas lieto CLI, ir pārliecinātāki par sevi un jūt, ka viņi dod komandas datoram, nevis tam pakļaujas. Turpretī GUI lietotāji ir nedroši. Tas varētu būt saistīts arī ar to, ka, lai lietotu CLI, ir jāapgūst paliels zināšanu apjoms, salīdzinot ar GUI, kas parasti ir ļoti intuitīvs – piesēdies un strādā, lai gan, protams, bez apmācības (arī pašapmācibas) nav iespējams lietot nevienu datorsistēmu. CLI mīnuss ir jau pieminētā vajadzība pēc padziļinātām zināšanām.

UNIX filozofija

Tagad pastāstīšu, kā tiek panākts tas, ka, strādājot konsolē, ir iespējama augstāka lietderība, nekā grafiskajā vidē.

UNIX filozofija ir normu kopums, kas balstīts uz vadošo UNIX OS programmētāju pieredzi. Tās stūrakmens ir teiciens "Dari tikai vienu un dari to labi!" (*"Do one thing, do it well."*), Doug McIlroy, UNIX programkanālu (*pipes*) izgudrotājs savā darbā UNIX filozofiju ir formulējis šādi: *"This is the UNIX philosophy: / Write programs that do one thing and do it well. / Write programs to work together. / Write programs to handle text streams, because that is a universal interface."* (Šī ir UNIX filozofija: / Raksti programmas, kas dara vienu lietu un dara to labi. / Raksti programmas, kas spēj sadarboties. / Raksti programmas, kas spēj apstrādā teksta plūstmas, jo tā ir universāla saskarne.)

Mūsdienās UNIX filozofiju definē 9 priekšraksti, no kuriem mums, apgūstot pamatus darbā ar komandrindas saskarni, būtiskākie ir:

- *Small is beautiful.* (Jo mazāk, jo skaistāk.)
- *Make each program do one thing well.* (Liec katrai programmai darīt vienu lietu labi!)
- *Choose portability over efficiency.* (Pārnesamība pāri efektivitātei.)
- *Store data in flat text files.* (Glabā datus teksta datnēs!)
- *Make every program a filter.* (Lai katra programma būtu filtrs!)

Tāpat bieži vien tiek sekots arī 10 citiem principiem, taču tie neskaitās filozofijas sastāvdaļa:

- *Use lowercase and keep it short.* (Lieto mazos burtus un īsus nosaukumus!)
- *Silence is golden.* (Klusēšana – zelts.)

Tā visa rezultātā mēs iegūstam daudz mazu un ātru programmu. Kuras iespējams savienot ar programkanāliem, bet izvadu vai ievadu nodot vai saņemt ar virzienmaiņas (redirect) palīdzību.

Piemēram, ja mums ir programmas, kas māk izvadīt kaut ko uz ekrāna, programma, kas māk saskaitīt divus skaitļus un programma, kas māk izvilkt kvadrātsakni, tad mēs varētu tās apvienot šādi: `izvadit "4 6" | saskaitit | kvadratsakne -a > rezultats.txt`. Tā kā programmas rakstītas, ievērojot UNIX filozofiju, pirmā programma, palaista vienatnē, uz ekrāna izvadīs skaitli 4, atsarpe, 6 un beigs darbu. Tā vietā mēs šīs programmas izvadu padosim nākāmās programmas ievadā, izmantojot programkānāla simbolu (*pipe*). Nākamā programma – saskaitit – paņems ievadu un izvadīs 10, pēc kā uzreiz beigtu darbu, taču tās izvads tiek padots nākamajai programmai, kas nolasīs 10 un izvadīs uz ekrāna 3 (pieņemot, ka mūsu programma, ja tai padots slēdzis -a, izvada tikai rezultāta veselo daļu). Kad programma kvadratsakne beigs darbu, izvads tiks ierakstīts datnē rezultats.txt. Līdzīgi – ar simbolu "<" – iespējams ielasīt programmas ievadu no datnes.

Protams, reālajā dzīvē programmas nav sadalītas tik smalki. Visas aritmētiskās operācijas veic viena programma, taču ir atsevišķas programmas, kas māk:

parādīt n pēdējās izvada rindiņas;

parādīt tikai rindiņas, kas satur noteiktu frāzi;

aizvietot simbolu virknes ar citām;

utt.

Kā?

Pirmkārt, konsolē var tikt ar taustiņu kombināciju Ctrl+Alt+Fn, kur n ir konsoles numurs. Parasti ir pieejamas sešas konsoles (1 – 6), kurās var darboties paralēli, starp tām pārslēdzoties ar kombināciju Alt+Fn. Tikt atpakaļ uz grafisko vidi var ar taustiņu kombināciju Ctrl+Alt+F7.

Iespējams palaist arī virtuālo termināli – kosoli, neizejot no grafiskās vides. Virtuālie terminālļi ir programmas, kuru vārdi parasti beidzas ar "term" un atrodami kaut kur galvenās GUI čaulas izvēlnē. Palaižot virtuālo termināli jāpiesakās (*login*) sistēmā nebūs – automātiski tiks paturētas privilēģijas, ar kurām strāda X sistēma. Reālajā konsolē būs jāievada pieteikumvārds (*login name*) un parole, pirms varēs piekļūt komandrindai.

Arī CLI, tāpat kā GUI, ir pieejamas vairākas čaulas. Viena no populārākajām Linux CLI čaulām ir *Bash* (*Bourne-again shell*), kas attīstījusies no *Bourne shell*, tāpēc stāstīšu par to.

Komandrindas uzvedne visdzrīzāk izskatīsies šādi: `user@niceone:~$`. Tā sastāv no vairākām daļām. Lietotājvārds, @, datora vārds, :, tekošā mape, lietotāja darbības režīms ($ – parasts lietotājs, # – priviliģēts lietotājs).

Lielākā daļa Linux komandu ir izpildāmas formātā "komanda [slēdži] [objekti]", piemēram `ls -l /etc/` vai `ls /etc/`, vai ls -l, vai `ls` (visas šīs komandas parādīs mapes saturu, taču katras komandas izvads būs atšķirīgs). Jebkurā brīdī iespējams nospiest tabulācijas taustiņu, lai liktu datoram automātiksi pabeigt komandas vai datnes nosaukumu.

Bash satur vairākas iebūvētas konstrukcijas: if ... then .. fi, while ... do ... done, for ... do ... done u.c.

Vienkāršāko komandu (mazo programmiņu, kas rakstītas atbilstoši UNIX filozofijai) saraksts (tās nav atkarīgas no čaulas):

ls [mape] – parāda mapes saturu.

cd [mape] – nomaina tekošo mapi.

cp <datne> <uz_datni> – kopē datni.

rm <datne> – dzēš datni.

mkdir <mape>, rmdir <mape> – izveido un dzēš mapi.

mv <datne> <uz> – pārvieto vai pārsauc datni.

chown <īpašnieks> <datne>, chgrp <grupa> <datne> – maina datnes īpašnieku.

chmod <pieeja> <datne> – maina datnes pieejas tiesības.

man <komanda> – izklāsta informāciju par komandas parametriem.

echo <teksts> – izvada tekstu uz ekrāna.

cat <datne> – parāda saturu.

head <datne>, tail <datne> – parāda pirmās vai pēdējās rindiņas.

more [datne], less [datne] – ļauj apskatīt garu datni tā, lai tā ietilptu ekrānā.

wc – saskaita vārdus.

Un tā tālāk.

Komandu ir ļoti daudz. Te vēl dažas – interesantākās – no tām (to nozīmi iespējams izlasīt ar komandu `man <komanda>`): clear, reset, grep, cut, sed, awk, tr, sort, uniq, dd, file, xargs, history (arī !<*prefix*> un !<*n*>), diff, find, basename, strings, ps, kill, fuser, crontab, su, finger, df, date, time.

Piemēri

`rmdir ??` – izdzēš visas tukšās mapes, kuras ir tieši divus simbolus garas.

`find . -name "*.html" | xargs rm` –

atrod visas datnes, sākot meklēt no tekošā kataloga rekursīvi, kuras atbilst īpašībai "vārds atbilst maskai *.html , t.i. vārds beidzas ar .html jeb paplašinājums ir html". Tad padot sagatavoto datņu sarakstu uz programmu rm kā parametru (ar xargs palīdzību). Tādā veidā tiek izpildītas komandas rm <datne>, uz katru no atrastajām datnēm.

Rezultāts: tiek rekursīvi izdzēstas visas datnes, kuru datnes tips ir html.

`cat `cat adrese`` –

Akcentēšanas simboli (*grave accent* - `) tiek lietoti, lai pārvērstu kādas komandas izvadu par citas komandas parametru (līdzigi kā programkanāls uz xargs). Dotajā piemērā no datnes "adrese" tiek paņemts saturs un padots kā parametrs pirmajai cat komandai. Teiksim, ja datnē "adrese" būtu teksts "ff22". Tad ši komandu virkne uz ekrāna parādītu datnes "ff22" saturu.

`for x in `find */*`; do mv $x `echo $x | sed 's/\//-/'`; done` –

ir struktūra <mape>/<datne>, un uzdevums ir tik vaļā no mapēm, saliekot visas datnes vienā, formatējot nosaukumu, teiksim, <mape>-<datne>. Skripta pamatā ir viens cikls. Cikls uzstāda x uz katru no rindiņām, ko atgriež komanda `` `find */*` ``. Šī komanda izvada visas datnes, kuras atrodas vismaz vienas mapes dziļumā. Katrā cikla iterācijā datne tiek pārsaukta un pārvietota uz tādu, ko atgriež komandu kombinācija `echo $x | sed 's/\//-/'`. Šeit ceļš līdz datnei tiek apstrādāt ar programmu sed. Tās parametri nosaka, ka pirmo atrasto slīpsvītras (/) ir jāaizvieto ar defisēm (-). To, ka jāveic aizvietošana, nosaka komandas nosaukums parametros (s). Tad aiz slīpsvītras tiek norādīts, kas jāmeklē (\/).

Tā kā slīpsvītra tiek izmantota, lai atdalītu sed komandas parametrus, tad jāizmanto atsoļa sekvence (*escape sequence*), ja to vajag lietot kā parametru vai tā daļu. Tāpēc pirms tās tiek lietota atgriezensikā slīpsvītra.

Pēc tam nāk teksts, ar ko jāaizvieto atrastais, ja tāds tiek atrasts. Komanda tiek noslēgta ar slīpsvītru.

`last -5 lietotaajs | grep \( | cut -d \( -f 2 |cut -d \) -f 1 | awk -F: '{a+=$1*60+$2} END{print a}'` –

parāda pēdojās piecas lietotāja "lietotaajs" pietekšanās reizes, atlasa tās, kas satur iekavu (iekava tiek pierakstīta ar atsoļa sekvences palīdzību, jo tas ir speciāls simbols *Bash* interpretatorā). Pēc tam no katras rindiņas tiek atstāts tikai tas teksts, kas ir iekavās (šajā gadījumā šis teksts ir pieteikšanās ilgums – cik ilgi lietotājs ir lietojis sistēmu – formā <stundas>:<minūtes>). Beigās šīs rindiņas tiek padodas uz programmu awk, kas, par atdalītāju ņemot kolu, summē, kādu laiku lietotājs pēdējo piecu reižu laikā ir patērējis, lietojot sistēmu. Tas tiek darīts pie lokāla mainīgā pieskatiot stundas (pirmo vērtību, atdalot ar kolu) pareizinātas ar 60 minūtēm un minūtes (otro vērtību, atdalot ar kolu). Beigās lokālais mainīgais tiek izvadīts uz ekrāna.

Tā rezultātā uz ekrāna parādīsies lietotāja sistēmā pavadītais laiks minūtēs pēdējo piecu darbības sesiju laikā.