Komunikace s okolními počítači

Příkaz finger

Příkaz `finger` vám umožní získat informace o ostatních uživatelích vašeho systému nebo o uživatelích sítě Internet. Jméno příkazu nepochybně vzniklo jako zkratka s reklamním podtextem ve firmě AT&T.

`finger [-slpm] [`*`user`*`][`*`@machine`*`]`

Volitelné parametry v příkazu `finger` mohou být mírně matoucí. Pomocí příkazu `finger` můžete získat informace o lokálním uživateli (například „sam“), o jiném počítači (například „@lionsden“), informace o uživateli vzdáleného počítače (například „sam@liondsen“) nebo informace o lokálním počítači (neuvede se žádný parametr).

Příkaz `finger` má další zajímavou vlastnost. Pokud se pokusíte získat informace o uživateli, jehož jméno přesně neznáte, pokusí se příkaz `finger` najít jméno sám (zkouší různé kombinace založené na původně zadaném jménu). To znamená, že když například zadáme příkaz `finger Greenfield`, obdržíme zprávu, že účet sam existuje pro jméno Sam Greenfield.

```
# finger sam
Login: sam Name: Sam Greenfield
Directory: /home/sam Shell: /bin/tcsh
Last login Sun Dec 25 14:47 (EST) on tty2
No Plan.
```

```
# finger greenfie@gauss.rutgers.edu
[gauss.rudgers.edu]
Login name: greenfie In real life: Greenfie
Directory: /gauss/u1/greefie Shell: /bin/tcsh
On since Dec 25 15:19:41 on ttyp0 from tiptop-slip-6439
13 minutes Idle Time
No unread mail
Project: You must be joking!
No Plan.
```

```
# finger
Login Name Tty Idle Login Time Office Office Phone
larry Larry Greenfield 1 3:51 Dec 25 12:50
larry Larry Greenfield p0 Dec 25 12:51
/home/larry#
```

Čtení zpráv elektronické pošty

mail [*user*]

Program `mail` poskytuje poněkud těžkopádnou možnost číst zprávy elektronické pošty. Zadáte-li příkaz `mail` bez jakýchkoliv parametrů, objeví se vám následující hlášení:

```
# mail
No mail for larry
```

Předpokládejme, že chcete odeslat zprávu elektronickou poštou sami sobě, abyste si mohli vyzkoušet způsoby jejího čtení:

```
# mail larry
Subject: Frogs!
and toads!
EOT

# echo "snakes" | mail larry

# mail
Mail version 5.5. 6/1/90 Type ? for help.
"/usr/spool/mail/larry" 2 messages
>N 1 larry Tue Aug 30 18:11 10/211 "Frogs!"
N 2 larry Tue Aug 30 18:12 10/211
&
```

Příkazový řádek uvnitř programu pro čtení elektronické pošty je uvozen znakem ampersand („&”). Umožňuje specifikovat spoustu jednoduchých příkazů a po zadání znaku `?` a Enter zobrazuje stručnou nápovědu.

Základními příkazy pro práci s programem `mail` jsou:

t *message-list*	Na obrazovce se zobrazí zprávy uvedené v seznamu *message-list*.
d *message-list*	Zprávy uvedené v seznamu *message-list* se zruší.
s *message-list file*	Zprávy uvedené v seznamu *message-list* se uloží do souboru *file*.
r *message-list*	Odpověď na zprávy -program mail zahájí proces sestavování nových zpráv, které budou odeslány každému, kdo vám odeslal zprávu uvedenou v seznamu *message-list*.
q	Program mail se ukončí a uloží každou zprávu, která nebyla zrušena příkazem d do souboru mbox ve vašem domovském adresáři.

Jak vypadá seznam *message-list*? Skládá se ze seznamu čísel oddělených mezerami, nebo může být vyjádřen ve formě intervalu, tedy například 2-4 (což znamená „2 3 4“). Také můžete uvést uživatelské jméno odesílatele. Například `t sam` zobrazí všechny zprávy odeslané uživatelem sam. Jestliže se seznam *message-list* neuvede, zobrazí se vždy poslední zpráva.

Používání systémů vzdálenými počítači

`telnet` *`vzdálený_systém`*

Hlavní prostředek pro využívání vzdálených systémů založených na operačním systému Unix představuje program `telnet`. (V současné době se spíše používá program `ssh`, který na rozdíl od příkazu `telnet` umožňuje šifrovanou komunikaci se vzdáleným systémem.)

Používání programu `telnet` je velmi jednoduché:

```
# telnet lionsden
Trying 128.2.36.41...
Connected to lionsden
Escape character is ´^]´.
lionsden login:
```

Jak vidíte z následujícího příkladu, po zadání příkazu `telnet` budete vyzváni k přihlášení se ke vzdálenému systému. Uživatelské jméno lze zadat jakékoliv (ovšem heslo musíte znát správné) a pak je vám vzdálený systém k dispozici téměř stejně jako váš lokální systém.

Normální způsob ukončení programu `telnet` spočívá v odhlášení se, ale je také možnost zadat znak `Escape`, kterým je zpravidla CTRL+]. Tak obdržíte nový příkazový řádek uvozený řetězcem `telnet>`. Nyní stačí napsat `quit` a pak stisknout klávesu Enter - spojení se přeruší a program `telnet` se ukončí. Pokud si to rozmyslíte a nechcete relaci ukončit, stiskněte pouze klávesu Enter.

Přenášení souborů

`ftp` *`vzdálený_systém`*

Normální způsob přenášení souborů zprostředkovává v operačním systému Unix program `ftp`, což je zkratka pro „*file transfer protocol*“. Po zadání příkazu `ftp` budete vyzváni, abyste se přihlásili ke vzdálenému systému téměř stejným způsobem, jako v případě programu `telnet`. Pak se vám zobrazí speciální příkazový řádek.

Nyní máte k dispozici několik příkazů běžných v operačním systému Unix, jež můžete aplikovat v příkazovém řádku programu `ftp`. Například příkaz `cd` i zde slouží k přepínání se mezi adresáři a příkaz `ls` slouží k zobrazení seznamu souborů uložených v aktuálním adresáři.

Navíc máte k dispozici dva důležité příkazy: **`get`** a **`put`**. Příkaz `get` je určen k přenosu souborů ze vzdáleného počítače do vašeho lokálního počítače a příkaz `put` slouží k opačnému úkolu. Oba příkazy jako implicitní používají váš domovský adresář a lokální adresář na vzdáleném počítači (který můžete měnit prostřednictvím příkazu `cd`).

S používáním programu `ftp` je spojen jeden problém. Spočívá v rozlišení mezi znakovými a binárními soubory. Protokol pro přenos dat programem `ftp` je velmi starý a implicitně předpokládá, že přenášené soubory jsou textové. Aplikuje-li se tento implicitní přenos na binární soubory, pak budou s největší pravděpodobností po přenosu poškozeny. Před přenosem binárního souboru proto používejte příkaz `binary`.

Program `ftp` ukončíte příkazem `bye`.

Příklady

Příklad 1: V Bourne shellu napište script **bu** , který pošle do Vaší e-mailové schránky všechny soubory z adresáře `/etc`, které začínají řetězcem `host` .

Řešení

```
$cat > bu
#!/bin/bash
for i in  /etc/host*
do
     echo  \
"************************************************
          Soubor $i
************************************************"
     cat $i
done|mail $USER
```

Příklad 2: Script z příkladu 1 modifikujte tak, aby poslal do Vaší e-mailové schránky soubory, které uvedete v příkazovém řádku.

Návod:

Řádek `for` změňte na `for i in $*` nebo na `for i`

Příklad 3: V Bourne shellu napište script, který pošle do Vaší e-mailové schránky všechny soubory, které uvedete v příkazovém řádku při jeho spuštění. Skript připojí k zasílaným souborům příkazy, které dovolí zaslané soubory snadno z e-mailu nainstalovat.

Řešení

```
$cat bun
#!/bin/bash
echo '#!/bin/bash'
for i in $*
do
     echo "cat >$i <<End of $i"
     cat $i
     echo "End of $i"
done | mail $USER
$bun ~/soubor1 ~/soubor2
```

Soubory `~/soubor1` a `~/soubor2` *jsou odeslány do Vaší e-mailové schránky*.

Příklad 4: V Bourne shellu napište script, který uvědomí uživatele o příchodu nové pošty. Script bude spuštěn na pozadí. Při spuštění bude možné zadat z příkazového řádku interval kontroly poštovní schránky.

Řešení

```
$cat nmail
#!/bin/bash
t=${1-60}
x="`ls -l $MAIL`"   {místo $MAIL skutečný obsah prom MAIL!}
while :
do
     y="`ls -l $MAIL`"
     echo $x $y
     x=$y
     sleep $t
done|awk '$14 > $5 {print "Mate novou postu"}'
$nmail -60 &
```

Příklad 5: Napište program v jazyce C, který uvědomí uživatele o příchodu nové pošty. Program bude spuštěn na pozadí a bude v pravidelných intervalech (např. 60 sec) sledovat velikost poštovní schránky. Pokud zjistí, že schránka se zvětšila, bude předpokládat, že uživateli přišla nová pošta a vypíše o tom zprávu na standardní výstup.

Návod

Ke zjištění velikosti poštovní schránky použijte službu systému ***stat()***, která zjistí informace uložené v i-uzlu.

```
int stat(char *path, struct stat buf);
stat(path, &buf);
```

Služba do struktury `buf` přečte informace z i-uzlu souboru `path`. Velikost souboru bude v položce `buf.st_size`. Pokud čtení informací z i-uzlu skončí neúspěšně, funkce vrátí -1. Definice struktury je v hlavičkovém souboru `<sys/stat.h>`.

Řešení

```
#include <stdio.h>
#include <sys/stat.h>
#define MAILDIR "/var/spool/mail"
extern char *getenv();

main(argc, argv)
     int argc;
     char *argv[];
{
     struct stat buf;
     char *name;
     int lastsize=0;
     if ((name=getenv("USER")) == NULL)
          fprintf(stderr,"nemohu ziskat uzivatelske jmeno");
     if (chdir(MAILDIR) == -1)
               fprintf(stderr,"nelze otevrit %s",MAILDIR);
     for (;;){
          if (stat(name,&buf) == -1) /*neni postovni schranka*/
               buf.st_size = 0;
          if (buf.st_size > lastsize)
               fprintf(stderr,"\nMate novou postu\007\n");
          lastsize = buf.st_size;
          sleep(60);
     }
}
```

Šifrování hesla

Ukažme princip, pomocí kterého lze při znalosti tabulky uživatelů snadno dešifrovat krátká hesla. Kódování hesla provádí systémový program **passwd** a používá k tomu funkci ***crypt():***

```
char * crypt (char * key, char *salt)
```

kde `key` je pointer na kódované slovo (program **passwd** sem vloží pointer na zadané heslo) a `salt` je pointer na dva libovolné znaky (program **passwd** je zvolí před kódováním). Funkce vrací pointer na řetězec znaků, který obsahuje na začátku dva znaky `salt` a pak 11 znaků šifry zakódovaného slova. Kódované slovo musí být uloženo v poli typu **char** o velikosti 8. Pokud je kódované slovo kratší než 8 znaků, volné posice musí obsahovat 0 (znak `\0`).

Program **passwd** ukládá všech 13 znaků, které funkce *crypt()* vrátí, do tabulky `/etc/passwd` **.** Samotné heslo může být maximálně 8 znaků dlouhé. Pokud je delší, jsou ostatní znaky programem **passwd** ignorovány. Různé verze programů **passwd** většinou požadují, aby minimální velikost hesla byla 5 znaků a aby se mezi znaky vyskytoval alespoň jeden nealfabetický znak.

Pokud toto víme, snadno vyřešíme následující příklad.

Příklad 6: Tabulka `/etc/passwd` obsahuje následující řádku:

```
novak:J63vt8rtrZ0Po:10:101:Pavel Novak:/home/novak:/bin/bash
```

Má uživatel novak heslo vltava77?

Řešení

Úlohu vyřešíte následujícím programem

```
#define SALT "J6"
char *crypt();
main()
{
char h[8] = {'v','l','t','a','v','a','7','7'};
printf("zakodovane heslo=%s\n",crypt(h,SALT));
}
```

Pokud je tento program přeložen a spuštěn, vytiskne šifru hesla `vltava77`. Tuto šifru potom porovnáme s druhou položkou uživatelského účtu `novak` a pokud jsou oba řetězce shodné, má uživatel `novak` heslo `vltava77`.