

Milan Šimánek
simanek@fel.cvut.cz
unix.fel.cvut.cz/unix

<http://pitboul.wz.cz>

cd ~ přepnutí do dom. adresáře, nebo jen cd
ls -l dlouhý výpis

čas souboru - čas směny
- čas post. předupr
- čas směny právo

Is -a i soub. kãt lãhon

mm Coulton

pr. dobroj cil

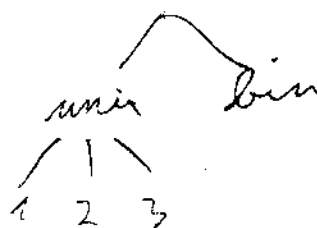
alias pyrite nodaveri aliasii

echo aboj vyprse reberce

echo \$PATH

mkotir -p unix/1 unix/2 unix/3 bin

vyhovět najeďnor



undir

malé souboj - nastavi souboj. aktuální čas,
jinak ho vysloví s 0 delkou

um -rf adiret

rekursivne na nje se ne deli

placit šejna oslabné

$$= |x| \cdot |x| = x^2$$

die | dem | spürten
hais

u adresáře x dovoluje
do něj přepnout

chmod 200 unix

! práva v osmičkové soustavě

chmod u + x
g - w
o = x

user/group/others

přidal
odebral
nastavil

chmod u+x unix

chmod ug=rwx unix

⑤

jednoduše popsat, nijaké slož. použití
data pro vypracování publikum
referát a semestrálka stejného téma
refe. určen nejpozdější termín

Terminál

slzy vlastnosti terminálu (rozdruhu)

escape sekvence

/etc/termcap - obsahuje informace o všech terminálech,
co je mu potřeba poslat, aby udělal
určitou věc

řekneš přikaz banner

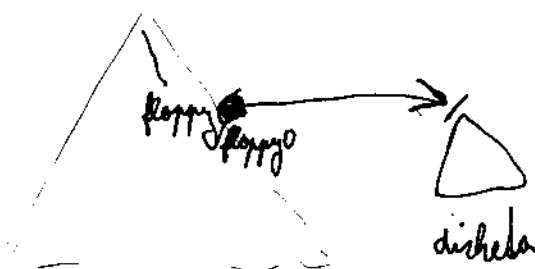
editor vi

funguje i bez nastaveného TERM

→ vypisuje aktuální řádek :) řádkový režim,
umí všechno, ale je vidět jen jeden řádek

Práce se soubory

/ všechno v jednom stromě



soubor umístěn /adr1/adr2/soubor...coholiv...krome...komitka
v názvu nemusí být /, jinak pravdu coholiv

msdir

rmdir

cp -n o o

rekursivně (adresář)

ls

cp

mv

rm

rm -r o rekursivně

ls ignoruje standardní soubory a se jm. řádk. soubory .jmeno
ily sáz normálně se neobslužovaly

cd

file ~~so~~ soubor - určí, co je soubor sác podle obrázků

symbolické linky

ln -s /etc/hosts H

hardlink

více ~~so~~ 'souborů' ukazují na jeden fyzický soubor

ln něco link

⑥

referat sort - řazení

uniq - označuje nebo filtruje opakuje se řádky
(jen když jsou se sebou - ~~průměrná~~ přeměna)
mu výstup se sortu

find

Hledání s * (treba *.jpg) tedy uzavřít
do apostrofu, jinak to shell může
nahradit případným existujícím souborem

' '

mezi apostrofy se upravuje

" "

funguje \, \$, ' - otevírací apostrof

\ ochrání následující znak

echo a _ " _ ' _ &

✓ právní parametry

rm find . -name '*.bak'

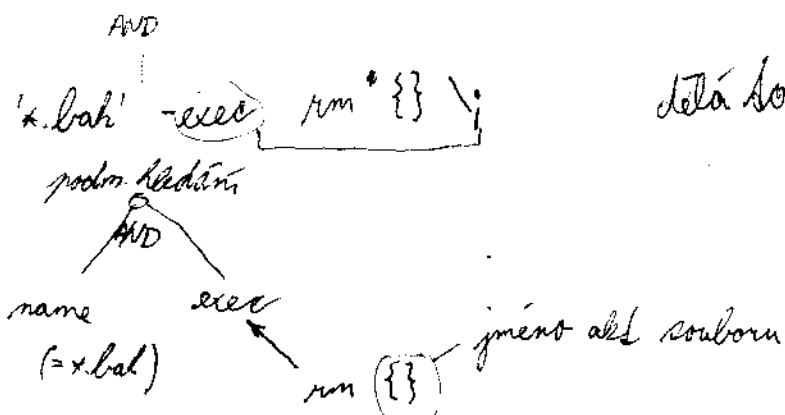
- nejprve vezme find,
co ptá, až jako
a potom come rm

- rm jednoduše soubor
param
- pouze pro málo souborů

rm *.bak

rm -r *.bak

find . -name '*.bak' -exec rm {} \;



tedy to samé, jako

napřed vyhodnotí jen první část, když jméno neodpovídá, dál už nevyhodnocuje

rm hodnekrát s jiným param

(7)

- referát GREP
moon.felk.cvut.cz/~chalup1

- referát - Přístupová práva

umask - práva nov. souborů

F 666 AND NOT xyz

D 777 - - -

umask xyz

vládní řádem procesu,
dělá se

(8)

referát - filter awk

referát - # ps
umask
time

http://logout.sk.cvut.cz/
~drahosk/referat.html

referát - signály

time

real - skutečný čas běhu

user - čas, který běžel jako proces

sys - čas, který pracovalo jádro pro tento proces

user + sys může být větší než real - dělá na více procesorech

~~veliký~~ vysoký čas user, pokud nemá něco počítat,
znamena nejjednodušší aktivní čekání

ps

proces zombie

zabývá pouze PID

- už skončil, čeká se, až si rodič přečte

návratovou hodnotu, fyzicky neexistuje

rodič upozorněn signálem

uchován, aby

neomylel vnítknout

nový proces se stejným PID,

a pokud do té doby není hodnota

vývedena



kill

SIGKILL

- končení procesu při pádu spájení,
návratu terminálu

0 - signál neexistuje, funguje v rámci procesu,
může se při aktivaci

Je třeba mít
dobrou nádobu

(9)

referat - rozšíření shell ~~více~~ při testování
podmínky

mate.no-ip.com/unix

přihaz - test

na račíteln skriptu

`#!/bin/sh` m
 └──┬──
 jméno
 interpretu

komentář

shell ignoruje

ale využije jádro při spuštění

pozná, že jde o interpretovaný skript.

zavolá správný shell

skript pro awk

`#!/bin/shawk -f` , nastaví práva spuštění
a bude to fungovat

ale i pro jiné, # musí být shodný s tím skriptu

no `awk '{print $3}'` vypíše 3. sloupec ze stol. sloupů
včetně pomocí awk skriptu

nakly podle sebe

if
then ① ^{ls} _{ps} — rozhoduje se podle úspěšnosti
posledního příkazu

else ②

fi ③

3 bloky příkazů

if ls/grep ahoj > /dev/null
then

rozhodne podle nabzení ahoj

||

ls/grep ahoj

if [\$? = 0] — časové složitější
then

ls přáť na jeden řádek, pak se oddělí středníkem

if [-f "\$1"]

— když není proměnná jasně
definovaná, URČITĚ do rozvozu,
jinak se může stát cokoliv

1) `[-f /etc/hosts] && [ ]`

2x se pustí test,
se andujou

návratové hodnoty

2) `[-f m -a m ]` spustí se jen jednou

moderní shelly přímo interpretují,
přímě spouští nový proces test,
⇒ by bylo pomalejší

ksh ravedl `[ ]`

ksh a výp

ksh

`[ ]`

↕

test

(zpětná kompat.)

`[[ ]]`

string

`[[ "$A" = *.txt ]]`

proměnná vyhovuje
vzoru

`[ ]` by nahradil
vyhov roaborama

`(( ))`

int

`(( abc = a + b + c ))`

`if (( A > B ))`

then

TADY neverme
jako přeměnování

skript a
~~a~~ x=10
 export x
 b

skript b
 echo \$x



exportovaná prom. do prostředí
 spustí se do volaného podprocesu,
 spadly se

~~a~~ `./b` spustí v 'aktuálním procesu'

Kávočky v shellu

```
{
ls
ps
} | more
```

výstup celé kávočky přesměrován
 na more

Reprát procedury v shellu

spuštění skriptu

sh [volby] skript [params] lze vybrat shell

./skript použije shell uvedený na řádce

proměnné ^{environ} ~~skriptu~~ HOME, PATH, USER každý proces má kopii

vnitřní prom. shellu \$0 jméno prog.
 \$* seznam param
 \$#

kolona - posloupnost příkazů příkaz1|příkaz2|...|příkazN
celkový náv. kód podle příkazu

sexnam - posloupnost kolon, oddělené " & && ; - schvácené
ukončené & ; ↵
na pozadí data příkazů
příkaz stane chybou pokud předchozí úspěšný

sexnamy lze uzavřít (), {}

- () provádí v kopii shellu - nový proces
- { } v akt shellu - pouze slouží příkazům,
ale kterýkoliv příkaz může vynutit
spuštění subshellu

B = 20

export B

vypíše

A = 10

(echo \$A \$B)

10 20

A = 1; B = xx

)

echo \$A \$B

10 20

do závorky projdou všechny proměnné,
provede se pouze fork(), ten
nevylze nic

když se volá jiný skript, tak fork(), pak exec() →
dostane pouze exportované proměnné

if grep 'A:' /etc/passwd > /dev/null

then

echo uzivatel A existuje

else

echo uzivatel A neexistuje

fi > soubor

celý příkaz if se přesměruje
do souboru

v tomto podobně jako {...}

read L - přečte 1 řádek ~~z~~ do L,
read L < soubor - přečte 1 řádek ~~z~~ souboru
práci načíst

i=1

while read L

do

echo "\$i \$L"

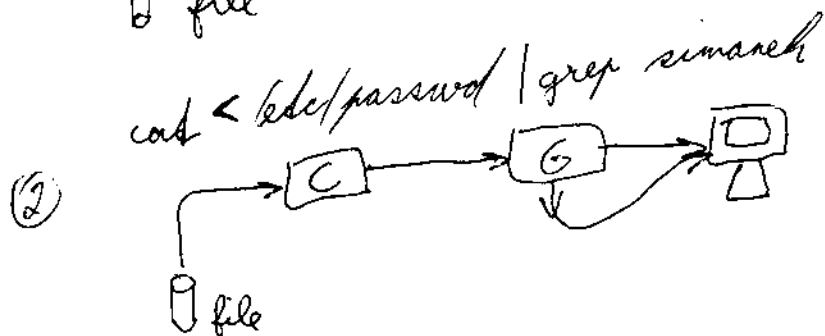
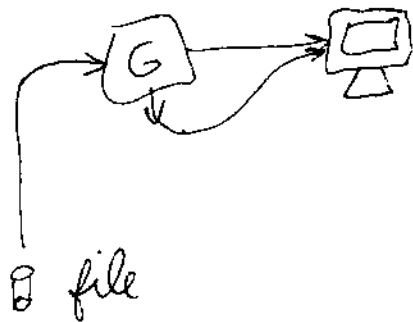
let i=i+1

done < /etc/group

echo "CELKEM \$i"

spočítání počtu usergroup
(řádků v souboru)

① `grep simanek < /etc/passwd`



!
 ①
 upravený skript

```

cat /etc/group | while read L
do

```

```

done

```

vynuceno rovnou
pro while vznikne
podproces, i když
to není vidět

→ prom se nedostane ven

častá chyba!
nesnadno objevitelná

Funkece a procedury v shellu

skript

```
abc()
```

```
{
```

```
ls $1
```

```
ps
```

```
A=2
```

```
}
```

telo procedury

```
A=1
```

```
abc p1 p2 > soubor
```

mozno

```
echo $A
```

- vypise 2 - proc spustena v
abst. shellu,
pokud není vynuceno
obalnostmi

treba abc p1 p2 | more

```
abc
```

```
{ xyz
```

```
}
```

```
xyz
```

```
{ abc
```

```
}
```

← je treba, aby bylo definovano az,
coz uz je

```
abc
```

11

8

sed -- a b c

↓ ↓ ↓
\$1 \$2 \$3

rozeteli podle odobrovane

isat mrit v ede vsechna fyz jména pro lp

referat - kalohování