A7B38UOS Úvod do operačních systémů

7. Cvičení

Příkaz find, práce s procesy a úlohami, plánování úloh

Find

find adresář [výraz]

V daném adresáři hledá soubory podle zadaného výrazu.

Výraz se vyhodnocuje zleva doprava a vyhodnocení končí v okamžiku, když je jasné, že výraz není splněný.

Implicitně jsou jednotlivé části výrazu spojeny logickým součinem.

Jednotlivé části můžeme spojit logickým součtem –o.

Jednotlivé části výrazu můžeme seskupovat pomocí závorek \(a \).

Také můžeme využít operátor negace \!.

Find

Ve výrazu můžeme použít následující příkazy:

-print	***výpis souborů, které splňují výraz***
-ok příkaz {} \;	***interaktivní provedení příkazu pro soubory, které splňují výraz***
-exec příkaz {} \;	***neinteraktivní provedení příkazu pro soubory, které splňují výraz***

Ve výrazu můžeme použít následující podmínky pro:

-type [d,f,l,b,c]	***typ souboru***
-inum n	***číslo i-uzlu***
-name ´vzor´	***jméno (lze použít znaky *,?,[ ] ,...)***
-perm nnn	***přístupová práva***
-mtime [n | -n | +n]	***čas modifikace***
-atime [n | -n | +n]	***čas přístupu***
-ctime [n | -n | +n]	***čas vytvoření***

Find

Příklady:

find $HOME

find $HOME –type f

find /bin –type f –name at

find /bin –type f –name at

find /usr/bin -type f -name ´*grep´

find . -atime -1

find . -name ´[A-Z]*´

find . \! -name ´[A-Z]*´

find . -name ´f*´ -print

find . -name ´s*´ -print

find . \(-name ´f*´ -o -name ´s*´ \) -print

find . \! \(-name ´f*´ -o -name ´s*´ \) -print

find . -print -name ´f*´ -print

find . -name core -ok rm {}\;

find . -name core -exec rm {}\;

Find

Nalezení všech souborů v adresáři /home/predmety/uos/common

```
dir=/home/predmety/uos
find $dir
```

Nalezení všech obyčejných souborů

```
find $dir -type f
```

... větších než 1000 bloků

```
find $dir -type f -size +1000
```

... menších než 100 bajtů (znaků)

```
find $dir -type f -size -100c
```

Find

Nalezení všech obyčejných souborů mladších než týden

find $dir -type f -mtime -7

... starších než 10 dní

find $dir -type f -mtime +10

... s alespoň 1 dalším hardlinkem

find $dir -type f -links +1

Nalezení všech souborů s nastaveným set-gid bitem

find $dir -perm -g+s

Find

Vypsání detailů nalezených souborů

find /usr/bin -name '*grep' -ls

Spuštění externího příkazu

find /etc -type f -exec grep -l 'Debian' {} \;

Spuštění externího příkazu s dotazem

find ~ -type f -size 0 -ok rm {} \;

Find

Spočítejte kolik je obyčejných souborů v adresáři /usr (včetně podadresářů), které vlastní root ve skupině root, majících nastavený set-uid bit a jsou spustitelné obyčejným uživatelem (který není ve skupině sys). Jejich seznam včetně přístupových práv uložte do souboru prog.suid . Chyby zahoďte.

```
find /usr \
-type f \
-user root \
-group root \
-perm -u+s,o+rx \
-ls \
2>/dev/null \
| awk '{ print $3,$11 }' \
| tee prog.suid | wc -l
```

Find

Nalezení všech adresářů v aktuálním adresáři

find . -type d

Nalezení všech adresářů v aktuálním adresáři do hloubky

find . -depth -type d

... maximálně hloubky 3

find . -maxdepth 3 -type d (GNU find)

... minimálně hloubky 3

find . -mindepth 3 -type d (GNU find)

Proces

- **Proces je spuštěný program.**
- **Každý proces má v rámci systému přiřazeno jednoznačné číslo procesu PID.**
- **Každý proces má svého rodiče a zná ho pod číslem PPID.**
- **Proces vzniká použitím systémových volání: *fork()***

vytvoří nový proces, který je kopií procesu, z kterého byla tato funkce zavolána

v rodičovském procesu vrací funkce PID potomka (v případě chyby -1)

v potomkovi vrací funkce 0

nový proces má jiné PID a PPID, ostatní vlastnosti dědí (např. EUID, EGID, prostředí, ...) nebo sdílí s rodičem (např. soubory, ...)

kódový segment sdílí potomek s rodičem

datový a zásobníkový segment vznikají kopií dat a zásobníku rodiče

Proces – identita procesu (RUID, RGID)

- systém si pamatuje pod jakým uživatelským účtem jsme se původně přihlásili (např. lokálně nebo vzdáleně pomocí ssh,...)
- Reálnou identitu procesu lze zobrazit např. pomocí následujícího příkazu

ps -o ruid,rgid,comm

- Efektivní (aktuální) identita procesu (EUID, EGID)

slouží k autorizaci procesu uvnitř systému (např. při vyhodnocování přístupu k souborům, ...)

při přihlášení je reálná a efektivní identita totožná

efektivní identitu lze změnit např. pomocí příkazu su nebo speciálních práv binárních souborů

lze ji zobrazit např. pomocí následujícího příkazu

ps -o uid,gid,comm

Vlákno (Thread)

- Vlákno je spuštěný podprogram v rámci procesu nebo jádra.
- Proces tvořený více vlákny může být vykonáván na více procesorech současně.
- Klasický jednovláknový proces může být vykonáván vždy jen na jednom procesoru.
- Vytvoření nového vlákna je rychlejší než vytvoření nového procesu.
- Nové vlákno může vzniknout např. pomocí knihovní funkce pthread_create().

Změna identity procesu

- ***Identitu procesu nastavuje kernel při startu procesu nebo ji mění na žádost procesu.***
- ***Obvykle jsou RUID a EUID resp. RGID a EGID stejná a dědí se od rodičovského procesu.***
- ***Ve zvláštních případech se nedědí, ale nastavují se všechna nebo jen některá:***

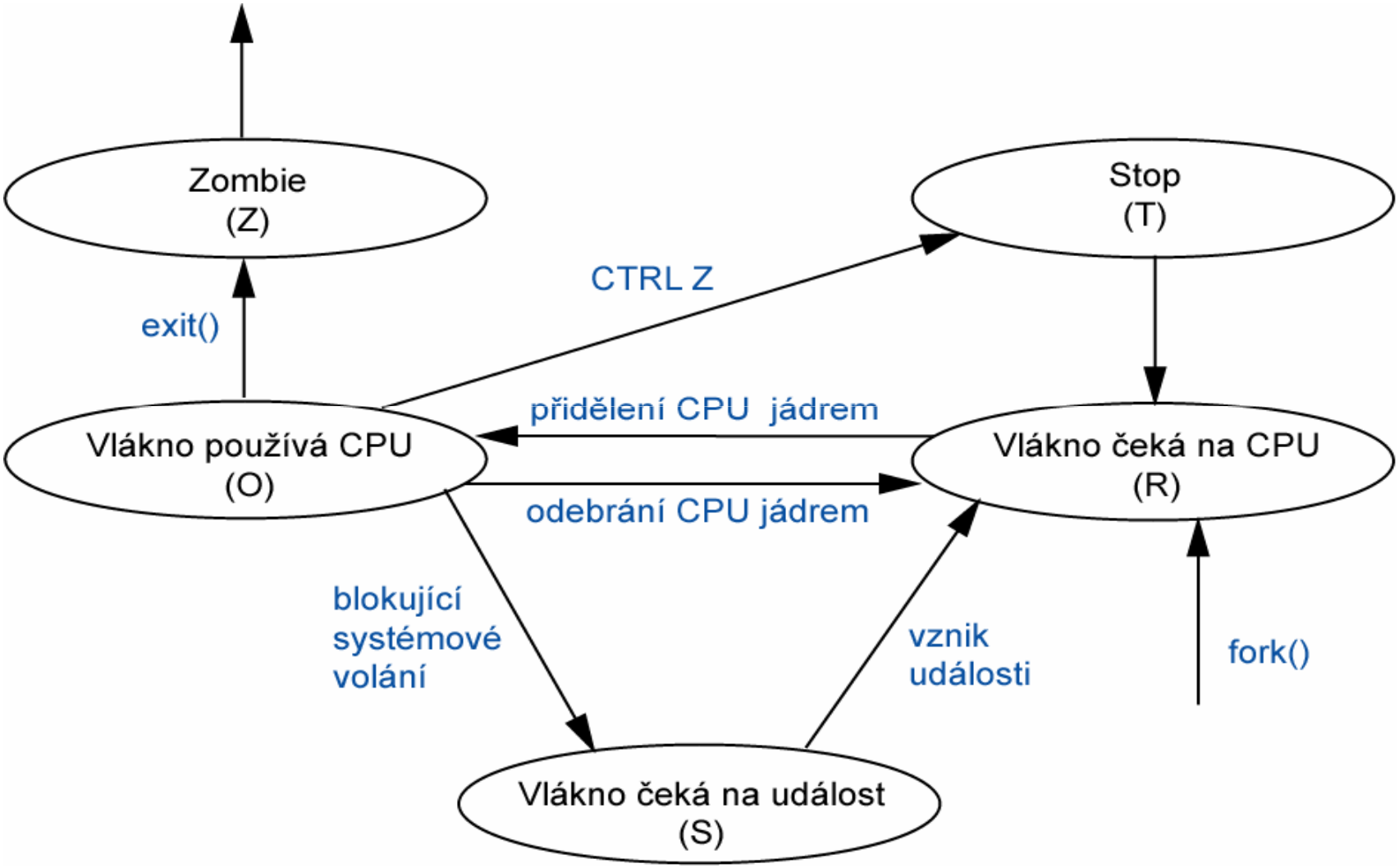
při přihlášení (pomocí procesů login)

pomocí příkazu su

u binárních programů s nastaveným suid bitem se mění EUID

u binárních programů s nastaveným sgid bitem se mění EGID

Ze života vlákna/procesu



su

su [-] [uživatelské jméno]

- ***Startuje nový shell pod novou identitou.***
- ***Původní shell nekončí, po odhlášení z su se v něm pokračuje.***
- ***Je-li su volán uživatelem, vyžaduje heslo, od roota ne.***
- ***Je-li uveden přepínač -, provede přihlašovací skripty (nastaví prostředí).***
- ***Je-li vynecháno přihlašovací jméno, doplní se jméno root.***
- ***Startuje nový shell s novou skupinovou identitou.***

sudo

sudo [přepínače] příkaz

- ***Umožnuje uživatelů spustit vybrané příkazy s právy roota.***
- ***Původní shell nekončí, po ukončení přikazu se v něm pokračuje.***
- ***Povolené přikazy pro jednotlivé uživatele nastavuje root.***

ps

ps

Vypíše krátkou informaci o procesech odstartovaných z daného terminálu.

ps -e ***nebo*** ps aux

Vypíše krátkou informaci o všech procesech v systému.

ps -f ***nebo*** ps -l

Vypíše detailnější informaci o procesech:

S	***stavy procesu (O, S, R, Z, T)***
PID, PPID	***číslo procesu a číslo rodičovského procesu***
PRI	***priorita***
NI	***NICE hodnota***
STIME	***čas spuštění procesu***
TIME	***čas spotřebovaný procesem na procesoru***
TTY	***terminál přidělený procesu***
CMD	***příkaz, který spustil proces***

ps

ps -o format

- ***Informace jsou vypsány ve formátovaném tvaru.***
- ***format může obsahovat následující zkratky:***

user ruser group rgroup uid ruid gid rgid pid ppid pgid sid pri
nice class time etime stime s c lwp ...

ps -Le

- ***Zobrazí informaci i o jednotlivých vláknech běžících v systému.***

ps

Výpis podrobností o procesech uživatele

passwd $USER &

ps -f -u $USER ***(effective)***

ps -f -U $USER ***(real)***

ps -ef | grep "^ *$USER "

Formátovaný výpis některých podrobností

ps -f -U $USER -o user=EFFECTIVE -o ruser=REAL \

-o pid,pcpu,comm

pgrep

pgrep [-lvx] [vzor -u seznam_uživatelů ...]

Vypíše PID (s –l navíc jména) procesů, které splňují zadané parametry (např. obsahují ve svém jménu daný vzor, běží pod identitou daných uživatelů,...).

S parametrem –v vypíše procesy, které nesplňují zadané parametry.

Vzor může obsahovat i regulární výraz.

pgrep

Zjištění PID procesů, jejichž jméno obsahuje bash

pgrep -u $USER bash

ps -u $USER | grep bash | awk '{print $1}'

Zjištění PID a jména procesů, které jsou spuštěny přímo z aktuálního procesu ($$ je pid rodičovského procesu)

sleep 100 & sleep 100 & sleep 100 &

pgrep -l -P $$

Zjištění doby běhu programu nebo úlohy

time du

time sleep 5

time { ls; sleep 5; }

pstree

Zobrazení stromu procesů (dědičných vztahů)

pstree

Modifikace priorit

Počáteční prioritu lze snížit (root může i zvýšit) příkazem:

nice –priorita příkaz

nice –n priorita příkaz

- ***kde priorita je číslo v rozsahu 1-19***
- ***větší číslo = větší snížení priority***
- ***záporné číslo = zvýšení priority (jen root)***

Modifikace priorit

Změna priority procesu

```
time echo "for (i=0;i<1000;i++) e(i)" \
| bc -l >/dev/null
time echo "for (i=0;i<1000;i++) e(i)" \
| nice -n 19 bc -l >/dev/null
```

Přenastavení speciálních znaků (^C → ^T) !!!Nezapomeňte obnovit původní stav!!!

```
stty -a | grep '\^'
stty intr ^t
stty -a | grep '\^'
stty intr ^c
```

Informace o průběhu procesu

```
strace date 2>&1 1>/dev/null | grep ^open
strace -t open date 1>/dev/null
```

Signály

- ***Signály umožňují přerušení procesu zvenku (např. jiným procesem, jádrem při špatném přístupu do paměti,...).***
- ***Každý signál je definován jménem a číslem.***
- ***V různých verzích Unixu se signály mohou lišit.***
- ***Seznam signálů lze vypsat příkazem*** kill -l
- ***Signál lze zaslat procesu pomocí příkazů:***

kill -signál PID

pkill -signál [-vx] [vzor –u seznam_uživatlů ...]

Signály

Některé signály lze zaslat procesu pomocí kombinace kláves:
(viz. stty –a nebo man stty)

Kombinace kláves	*Význam*
CTRL C	*Předčasné ukončení běžícího procesu. 2 SIGINT*
*CRTL *	*Předčasné ukončení běžícího procesu. 3 SIGQUIT*
CTRL S	*Pozastavení výstupu na obrazovku. 23 SIGSTOP*
CTRL Q	*Uvolnění pozastaveného výstupu. 25 SIGCONT*
CTRL Z	*Pozastavení běžícího procesu (ne u sh). 24 SIGTSTP*
	Nikoliv ukončení!!!

Signály

15 SIGTERM (TERMinate)

Posílán příkazem kill PID.

Standardní reakce je ukončení procesu. Obvyklý způsob předčasného ukončení procesu.

9 SIGKILL (KILL)

Standardní reakce (kterou nelze změnit ani ignorovat) je ukončení procesu.

Signály

1 SIGHUP (HangUP)

Posílán procesu, když končí jeho rodič (shellu, když zavěsil modem).

Standardní reakce je ukončení.

Končí-li proces, je signál poslán všem jeho potomkům. Proto při odhlášení jsou ukončeny všechny procesy spuštěné v rámci tohoto přihlášení.

Má-li proces pokračovat po ukončení rodiče, je třeba použít příkaz

nohup příkaz &

Některé procesy signál odchytávají a interpretují ho jako žádost o restart (např. init, named, sendmail, inetd, syslogd, automountd).

Reakce na signály

Každý proces má při startu definovanou reakci na všechny signály.

Obvykle je to „konec“ nebo „core a konec“.

Proces může reakci předefinovat kromě signálů KILL a STOP.

Reakci na signály můžeme měnit příkazem trap.

- ***Nastavení reakce na signál(y):***

trap ´příkazy´ signály_oddělené_mezerou

- ***Výpis nastavených reakcí:***

trap

- ***Ignorování signálů:***

trap ´´ signály_oddělené_mezerou

- ***Nastavení výchozí reakce:***

trap – signály_oddělené_mezerou ???

Reakce na signály

Nastavení reakce na signál

trap date INT

^C

trap 'rm /tmp/data.$$; exit' 2 3 15

trap -p

trap '' quit ; trap – int ; trap -p

Výpis signálů

kill -l ; kill -l 2 ; kill -l quit

Zaslání signálu procesu (signály 3 a 15). SIGQUIT neudělá nic, viz předchozí nastavení. SIGTERM (smazat soubor /tmp/data.$$ a ukončit bash) uspěje. Nelekněte se. Ukončíte si terminál.)

touch /tmp/data.$$; ls /tmp/data.*

kill -3 $$; kill $$

Popředí, pozadí

Na popředí

$ příkaz

- ***Proces přebírá vstup i výstup.***
- ***Příkaz (který není zabudovaný) je spuštěn jako nový proces. Shell čeká na jeho dokončení.***

Na pozadí

$ příkaz &

- ***Shell nečeká na jeho dokončení a nepředává mu vstup (pokud nebyl přesměrován).***
- ***Jestliže se proces pokusí číst z nepředaného vstupu je pozastaven.***
- ***Na výstup psát může. Pokud výstup nebyl přesměrován, míchá se s výstupem shellu.***

Mimo shell

$ nohup příkaz &

- ***Proces poběží i po ukončení shellu.***
- ***Výstup procesu je přesměrován do souboru nohup.out.***

Plánování spuštění

Pomocí příkazu at nebo crontab lze naplánovat spuštění procesu v daném čase.

Spuštění procesu provádí systémový démon cron.

Proces poběží explicitně pod identitou toho, kdo proces naplánoval a výstup bude poslán mailem.

Plánování spuštění

Nastavení a vypsání opakovaného spouštění úloh

export EDITOR=vi

crontab -e

**** * * * * $HOME/logger***

minuta
hodina
den
měsíc
den v týdnu

Možné hodnoty:

**** každá hodnota***

****/2 každá 2. hodnota***

5 hodnota 5

2,7hodnoty 2 a 7

1-5hodnoty 1,2,3,4,5

crontab -l

Správa úloh+

Umožňují všechny shelly kromě Bourne shellu (/bin/sh).

Každý proces, který je spuštěn v dané instanci shellu, má v této instanci přiřazeno jedinečné číslo úlohy (JID).

Pomocí JID se můžeme na proces odkazovat pomocí příkazů:

Jobs ***vypíše seznam úloh(procesů) běžících v tomto shellu***

fg [%JID] ***přesune úlohu na popředí***

bg [%JID] ***přesune úlohu na pozadí***

kill –signál [%JID] ***pošle úloze daný signál***

Není-li v příkazu uvedeno JID, uvažuje se poslední úloha.

Správa úloh+

Spuštění úlohy na pozadí / popředí

cmd=/home/courses/Y36UOS/common/cv08/proces.sh

$cmd & sleep 3 ; $cmd

Pozastavení úlohy na popředí

$cmd

^Z

Seznam spuštěných úloh

sleep 10 & sleep 15 & sleep 20

^Z

jobs

Správa úloh+

Přesun úlohy na popředí / pozadí

sleep 100 & $cmd &

jobs ; fg

^Z

jobs ; bg %1

Přerušení úlohy na popředí

ls -lR /

^C

Spuštění úlohy na pozadí běžící i po skončení rodiče

nohup find /usr -name '*zip*' 2>/dev/null &

less nohup.out