

SVEUČILIŠTE U MOSTARU
FAKULTET STROJARSTVA, RAČUNARSTVA I ELEKTROTEHNIKE
PREDDIPLOMSKI STUDIJ RAČUNARSTVA

OPERACIJSKI SUSTAVI

VJEŽBE

Nastavnik: prof.dr.sc. Sven Gotovac
gotovac@fesb.hr

Asistent: doc.dr.sc. Željko Šeremet
zeljko.seremet@fsre.sum.ba

MOSTAR, TRAVANJ 2024.

PROCESI



SADRŽAJ

- Pojam procesa
- Ispis procesa na sustavu
- Prekidanje procesa
- Odvijanje procesa u pozadini
- Odvijanje procesa nakon isključenja terminala



POJAM PROCESA (1)

○ Program

- Datoteka s izvršnim kodom na disku računala.
- Primjerice, `/bin/bash` je program

○ Proces

- Program koji se izvršava na računalu
- Dakle, proces uključuje i niz dinamičkih podataka koji se mijenjaju tijekom njegova izvršavanja a mogu se nalaziti pohranjeni na disku ili u radnoj memoriji



POJAM PROCESA (2)

- Svaki proces posjeduje jedinstveni identifikator
 - **PID** - process identifikator
- Proces koji je pokrenuo trenutni proces je njegov roditelj, svi procesi pokrenuti od trenutnog procesa su njegova djeca
 - **PPID** - parent PID
 - Iznimka je proces za kojeg vrijedi $PID=1$ kojega je stvorila jezgra operacijskog sustava tijekom inicijalizacije sustava



UNIX PROCESI

- Pokretanje novog procesa ostvaruje se pozivanjem **fork()** funkcije unutar trenutnog programa
 - **fork()** stvara kopiju procesa i vraća **PID** novonastalog procesa ili 0 ako je proces novonastali proces
- Svaki roditelj odgovoran je počistiti za djetetom :)
 - Roditelj čeka da dijete završi sa izvođenjem preko **wait()** sistemskog poziva



ISPIS PROCESA (1)

- Ispis procesa obavlja se s naredbom **ps**
 - **processor status**
 - ako se naredba pokrene bez argumenata i opcija ispisuje procese vezane uz terminal na kojemu je pokrenuta
- Omogućuje tri načina rada: Unix, BSD, GNU
- Zadatak
 - U man stranicama pogledati ispis svih procesa korištenjem UNIX i BSD načina



ISPIS PROCESA (2)

- Ispis naredbe **ps** sadrži sljedeće kolone

PID ID procesa

TTY TTY za koji je proces vezan

TIME Ukupno vrijeme izvršavanja

CMD Naredba bez argumenata



ISPIS PROCESA (3)

- Dodatne informacije o procesima ispisuju se korištenjem opcije **-f**

UID Vlasnik procesa

PPID Roditelj procesa

CMD Naredba s argumentima



ISPIS PROCESA (4)

- Korisne opcije **ps** naredbe
 - e** - ispis svih procesa na sustavu
 - f** - dodatne informacije o procesima
 - o** - zadavanje ispisa željenih informacija
 - **-sort** -sortiranje ispisa (po PID ako nije navedeno)



ISPIS PROCESA (5)

○ Primjeri

- ispis trenutnih procesa sa definiranim prikazom atributa

ps -eo pid, ppid, user, nice - -sort user

- detaljan ispis SVIH procesa, uključujući dretve
ps -eLf



ISPIS PROCESA (5)

- S argumentom **-u** dobijamo ispis resursa što ih troše procesi
- Kolone specifične za ovaj ispis
 - %CPU** Količina "potrošnje" procesora
 - %MEM** Količina zauzete radne memorije
 - VSZ** Virtualna veličina memorije
 - STAT** Stanje procesa u trenutku izvršavanja naredbe
 - START** Vrijeme kada je naredba pokrenuta



STANJA PROCESA (2)

- Moguće vrijednosti statusa procesa

R aktivni (running) proces

S spavajući (sleeping) proces (20 sekundi ili manje)

I besposlen (idle) proces (više od 20 sekundi)

T zaustavljen (stopped) proces

Z zombi proces (proces koji je završio, a zauzima zapis u tablici procesa)



DODATNE NAREDBE ZA ISPIS (1)

- **pgrep** pretražuje procese na temelju imena i drugih atributa
- Zadatak
 - Pretražite procese korisnika root
- **pstree** ispisuje stablo svih procesa na sustavu
- Zadatak
 - Provjerite odnose procesa pomoću naredbe **pstree**



DODATNE NAREDBE ZA ISPIS (2)

- Naredba **ps** ispiše trenutno stanje svih procesa (snapshot)
 - Naredba **top** omogućuje konstantan ispis
- Podrazumijevane vrijednosti naredbe **top**
 - Osvježavanje se obavlja svake 3 sekunde
 - Tipka **s** i upisivanje broja mijenja tu vrijednost
 - Procesi su poredani po korištenju procesora
 - S tipkom **M** poredak se vrši po potrošnji memorije, tipka **P** po korištenju procesora



SLANJE SIGNALA (1)

- Signal je programski prekid (interrupt)
- Kada proces primi signal obavlja neku akciju, obradu ili jezgra operacijskog sustava obavlja akciju
 - Po završetku obrade proces nastavlja s normalnim izvršavanjem
- Svaki signal ima svoj broj i skraćeno ime
 - Ime signala služi korisnicima, a brojevi jezgri



SLANJE SIGNALA (2)

- Slanje signala iz komandne linije obavlja se naredbom **kill**
 - Tipke Ctrl+C i slične također šalju signale procesima!
- Ako je zaustavljen roditelj i dijete će biti zaustavljeno
 - Svaki proces kao svog pretka ima **init** proces



SLANJE SIGNALA (3)

- Sintaksa

kill [-<broj signala>] -<PID>

- Popis svih raspoloživih signala se može dobiti opcijom **-l**

Broj	Ime	Značenje
1	SIGHUP	Završiti s radom (Hang up)
2	SIGINT	Prekidanje (Interrupt)
9	SIGKILL	Kill (ne može biti ignoriran)
15	SIGTERM	Programska terminacija (podrazumijevani signal)



SLANJE SIGNALA (4)

- Naredba **killall** služi za slanje signala na temelju imena za razliku od **kill** naredbe (PID)
- Zadatak
 - Terminirajte proces s $PID = 1$
 - Što se dogodilo?
 - Otvorite dva terminala. Terminirajte ljusku onog drugog.
 - Što se dogodilo?
 - Pošaljite signal broj 9 (KILL) ljusci drugog terminala.
 - Što se sada desilo?



MIJENJANJE PRIORITETA PROCESA

- Kod rezerviranja CPU vremena neki procesi imaju veći prioritet
 - Ali većina korisničkih programa ima isti priotitet
- Naredbom **nice** procesi se pokreću sa višim ili nižim prioritetom
 - **renice** mijenja prioritet postojećeg procesa ili grupe procesa
- Zadatak: Odaberite proces i promijenite prioritet!



POSLOVI

- Ljuske prepoznaju koncept poslova (jobs)
 - Kod pokretanja procesa iz terminala ljuska proces stavi u prednji plan (foreground)
- Proces je moguće prebaciti u pozadinu i u prednji plan ili zaustaviti

Ctrl+Z

bg

fg

jobs



NAREDBA `bg`

- Ako se proces nastavlja bez ikakvog ispisa na ekranu i bez i kakvih zahtjeva za unosom, koristimo naredbu **bg** da ga stavimo u pozadinu, gdje će se odvijati dok se ne završi
 - naredbu **bg** možemo koristiti tek nakon suspendiranja procesa naredbom `Ctrl+Z`
 - Posljedica `Ctrl+Z` je slanje signala `SIGSTOP` procesu
- Terminiranje ljuske terminira njezine procese



NAREDBA `jobs`

- Naredba **`jobs`** prikazuje nam procese koji se izvršavaju u pozadini
- Procesi imaju identifikator koji nema veze s PIDom
- Taj identifikator može se koristiti kod naredbe **`bg`** i sličnih



NAREDBA fg

- Kada je proces zaustavljen, ili se izvršava u pozadini, može se ponovno staviti u aktivan način rada naredbom **fg (foreground)**
 - puno naredbi koje rade s procesima prihvataju identifikator/oznaku posla (jobID) kao argument, pa tako i naredba **fg**
- Ako identifikator nije naveden podrazumijeva se zadnji proces s kojim je nešto napravljeno



KORIŠTENJE `bg`, `jobs` I `fg` (1)

- Pokrenuti editor `vi` te ga zaustaviti sa `^Z`
- Pogledati naredbom `jobs` aktivne poslove
- Pokrenuti još jedan proces programa `vi` te ga također zaustaviti sa `^Z`
- Ponovo upotrijebiti naredbu `jobs` i pogledati aktivne poslove
 - S oznakom `+` označen je zadnji proces s kojim je nešto manipulirano, a oznakom `-` označen je predzadnji



KORIŠTENJE `bg`, `jobs` I `fg` (2)

- Primjetite kako su svi procesi u stanju `Stopped`
 - S naredbom **`bg`** prebacite proces s `JID`-om 1 u pozadinu
 - Što se desilo? Da li se proces nastavlja izvršavati u pozadini?
- Naredba **`kill`** prihvaća i `JID`, ali ga je potrebno označiti s `%`
 - Pošaljite signal `TERM` vi editoru s `JID`om 1
 - Što se desilo? Stavite posao 1 u prednji plan (`fg 1`)



OPERATOR &

- Nije nužno pokretati program pa ga zaustavljati sa **^Z**
 - Možemo koristiti operator **&** kod pokretanja programa
 - Ako program zahtijeva unos ili ispis, zaustavlja se
 - to ne ovisi o tome kako je program pokrenut
- Da bi se program odmah pokrenuo u pozadini, jednostavno napišite **&** na kraju komandne linije



PROCESI, GRUPE PROCESA I SJEDNICE (1)

- Pokrenuti proces vezan je na terminal
 - Terminiranjem ljuske prekinut je i proces
- Svaki proces dio je grupe procesa
 - Grupa procesa je obuhvaća isto što i posao
 - Neke ljuske ne podržavaju poslove
- Svaka grupa procesa dio je sjednice
 - Ovakvo grupiranje omogućuje slanje signala svim željenim procesima istovremeno



PROCESI, GRUPE PROCESA I SJEDNICE (2)

○ Primjer

```
$ cat
```

```
hello
```

```
hello
```

```
^Z
```

```
[1]+  Stopped
```

```
$ ls | sort
```

- Identifikator grupe procesa (PGID) jednak je identifikatoru prvog procesu grupe
- Svi procesi povezani cjevovodom su dio iste grupe
- Svi procesi unutar ljuske su dio iste sjednice



PROCESI, GRUPE PROCESA I SJEDNICE (3)

- Slika otprilike pokazuje stanje prethodnog primjera
- Primijetite kako je moguće ne imati TTY postavljen

Session 100		Session 101			
Job 100		Job 101	Job 102	Job 103	
XTerm (100)		bash (101)	cat (102)	ls (103)	sort (104)
stdin: -		stdin: /dev/pts/0	stdin: /dev/pts/0	stdin: /dev/pts/0	stdin: pipe0
stdout: -		stdout: /dev/pts/0	stdout: /dev/pts/0	stdout: pipe0	stdout: /dev/pts/0
stderr: -		stderr: /dev/pts/0	stderr: /dev/pts/0	stderr: /dev/pts/0	stderr: /dev/pts/0
PPID: ?		PPID: 100	PPID: 101	PPID: 101	PPID: 101
PGID: 100		PGID: 101	PGID: 102	PGID: 103	PGID: 103
SID: 100		SID: 101	SID: 101	SID: 101	SID: 101
TTY: -		TTY: /dev/pts/0	TTY: /dev/pts/0	TTY: /dev/pts/0	TTY: /dev/pts/0

LITERATURA

- <http://www.linux.com/archive/feature/125977>
- <http://www.linux-tutorial.info/modules.php?name=MContent&pageid=84>
- <http://www.linuxhq.com/guides/SAG/x1826.html>
- <http://www.win.tue.nl/~aeb/linux/lk/lk10.html>
- <http://www.linusakesson.net/programming/tty/index.php>



PITANJA

