

23 Procesy v OS Linux, 1. část

Obsah hodiny



Obsahem hodiny je popis je procesů z pohledu OS Linux: vznik, hierarchie a identifikace procesů.

Cíl hodiny



Po této hodině budete schopni:

- zjistit informace o procesech
- orientovat se v příkazech pro práci s procesy
- orientovat se ve vlastnostech procesů

Klíčová slova



PID, PPID, Signály, Hierarchie procesů, Proces init, Identifikace procesů, vznik procesů, Priorita procesů

23.1 Vznik procesu, hierarchie procesů

Procesy v systémech unixového typu vznikají pomocí volání jádra `fork()`, které způsobí jakési rozdvojení procesu. Proces, který volání jádra `fork()` vyvolal, se nazývá rodičovský proces (parent). Pokud `fork()` proběhne úspěšně, vzniká nový proces - tzv. potomek (child). Z toho plyne, že každý proces tedy má svého rodiče a vzniká tak hierarchie procesů. Každý spuštěný proces zná PID svého rodiče.

V hierarchii funguje jednoduchá komunikace prostřednictvím signálů. Např. zabitím rodiče se distribuuje signál dětem a ty buď také skončí, nebo je převezme proces `init`.

Linux startuje jako proces `init`, který spouští služby a tyto služby spouští další procesy. (Při spuštění konzoly se spustí proces `putty`, `login` a následně proces `bash`, který může spustit další proces). Proces `init` je rodičem všech procesů, pokud proces osiří (přijde o rodiče a z nějakého důvodu nedostane signál o ukončení) stane se přímým potomkem `init`.

V Linuxu lze docela dobře hierarchii mezi procesy vysledovat. Lze ji vypsát pomocí příkazu *ps tree* nebo *ps -H*, *ps -axjf*.

```

1  8340  1688  1688  ?          -1 Rl    1000    0:22  gnome-terminal
8340 8341  1688  1688  ?          -1 S     1000    0:00  \_  gnome-pty-helper
8340 8342  8342  8342  pts/0      8342 Ss+   1000    0:00  \_  bash
8340 8442  8442  8442  pts/1      8465 Ss    1000    0:00  \_  bash
8442 8465  8465  8442  pts/1      8465 R+   1000    0:00  \_  ps axjf

```

```
$ ps tree -pu
```

```

├─gnome-terminal (8340,pm)─┬─bash (8342)─pstree (8583)
│                          └─bash (8442)─mc (8545)─bash (8547)
│                          └─gnome-pty-helpe (8341)
│                          └─{gnome-terminal} (8343)
├─go-home-applet (1860,pm)
├─gvfs-fuse-daemo (1768,pm)─┬─{gvfs-fuse-daemo} (1769)
│                          └─{gvfs-fuse-daemo} (1770)
│                          └─{gvfs-fuse-daemo} (1771)
├─gvfs-gdu-volume (1881,pm)
├─gvfs-gphoto2-vo (1922,pm)
├─gvfsd (1761,pm)
├─gvfsd-burn (1930,pm)
└─hald (487,haldaemon)─hald-runner (571,root)─┬─hald-addon-acpi (967,haldaemon)
│                                              └─hald-addon-cpuf (963)
│                                              └─hald-addon-gene (930)
│                                              └─hald-addon-inpu (971)

```

Obrázek 23-1: Hierarchie procesů – příkaz *ps -H*, *ps -axjf* nebo *ps tree -pu*

23.2 Identifikace procesu

Každý proces je identifikován číslem PID (Process ID), které procesu přidělí operační systém. Každý proces získá číslo o jedničku vyšší, než předchozí proces, pokud toto číslo již nepatří jinému procesu. Čísla procesů jsou unikátní a v daný okamžik neexistují dva procesy se stejným PID. PID je platný po celou dobu běhu procesu až do ukončení procesu

Stav procesu lze monitorovat. Pokud jádro obsahuje podporu souborového systému procfs, jsou informace o procesech v adresáři */proc/číslo_procesu/*.

Pro identifikaci procesů – se používají příkazy: *ps* a *top*

Příkaz *ps* bez parametrů vypíše pouze procesy uživatele, který tento příkaz spustil a pouze ty procesy, které byly spuštěny ze stejného terminálu jako samotný příkaz *ps*. Příkaz *ps* zobrazí statický obraz procesů v okamžiku spuštění příkazu *ps*. Má řadu parametrů, kterými řídíme:

- jaké informace o procesu chceme zjistit,
- jakou skupinu procesů chceme vypsát:
 - procesy vázané na uživatele,
 - procesy vázané na terminál.

```
$ ps -l
```

	F	S	UID	PID	PPID	C	PRI	NI	SZ	WCHAN	TTY	TIME	CMD
000	S		500	1012	1009	0	74	0	729	wait4	pts/0	00:06	bash
000	T		500	13500	1012	0	69	0	474	do_sig	pts/0	00:00	vi
000	R		500	12943	1012	0	78	0	769	-	pts/0	00:00	ps

Příkaz *top* nevypisuje pouze výpis aktuálního stavu procesů v systému, ale dokáže tento výpis dynamicky po určitých časových intervalech měnit. Zobrazuje v reálném čase změny stavu procesů, jejich "boj" o procesor, aktuální velikost paměti, které procesy alokují a spoustu dalších užitečných informací.

Defaultní časový interval změny výpisu je nastaven na 5 sekund, lze jej však změnit pomocí parametru *-d*. Příkaz je poměrně náročný na systémové prostředky a zmenšení intervalu výpisu bude náročnost na systémové prostředky ještě zvyšovat.

Příkaz *top* vypisuje v záhlaví celkové statistiky procesů (jak dlouho běží systém, kolik pracuje uživatelů, kolik je spuštěno úloh, v jakých stavech, kolik je obsazeno paměti, ...), dále se pak vypisují informace o jednotlivých procesech.

```
top -b top-10:02:37 up 69 days, 34 min, 7 users, load average: 0.00, 0.10, 0.15 Tasks: 127
total, 1 running, 125 sleeping, 1 stopped, 0 zombie Cpu(s): 11.2%us, 1.7% sy, 0.8%
ni, 84.3% id, 1.7% wa, 0.1% hi, 0.2% si Mem: 385836k total, 380732k used, 5104k
free, 42856k buffers Swap: 497972k total, 232456k used, 265516k free, 73372k
cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
7923	root	15	0	184m	86m	79m	S	2.0	23.0	632:56.88	X
27569	ondra	15	0	136m	69m	36m	S	2.0	18.4	73:56.43	firefox-bin
23790	ondra	15	0	2168	972	1964	R	2.0	0.3	0:00.02	top
1	root	16	0	1580	164	1424	S	0.0	0.0	0:05.97	init
2	root	34	19	0	0	0	S	0.0	0.0	0:01.03	ksoftirqd/0
3	root	5	-10	0	0	0	S	0.0	0.0	0:17.46	events/0

Obrázek 23-2: Výpis příkazu *top*

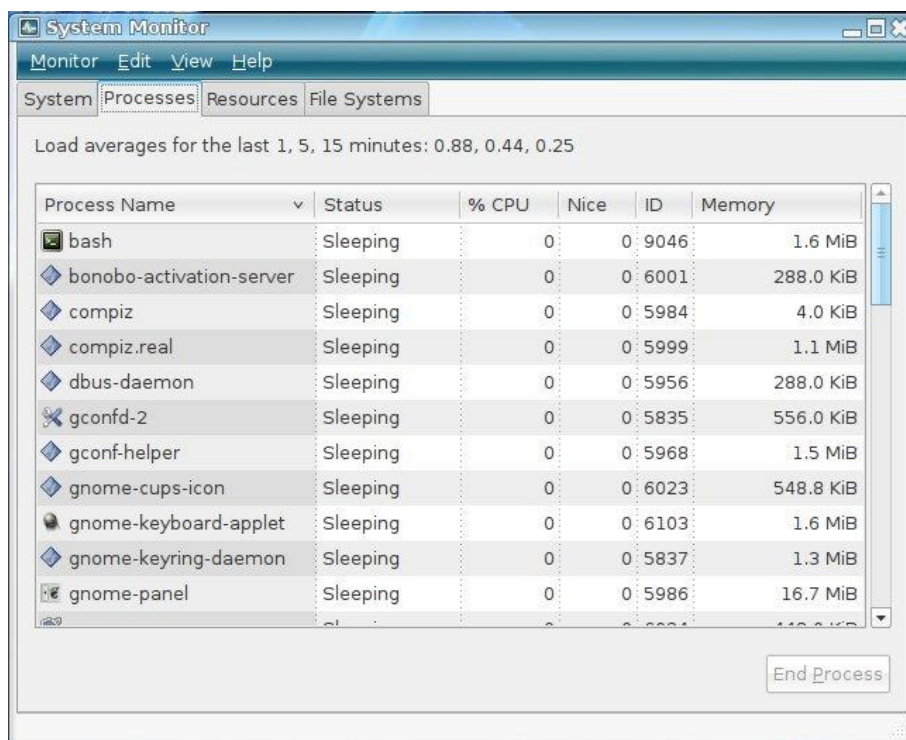
O procesech lze zjistit následující údaje:

- **PID** udává identifikační číslo procesu
- **USER** vypisuje identitu (uživatele) pod níž daný proces běží.
- **PRI** vypisuje aktuální výši priority daného procesu.
- **NI** udává výši priority zadanou příkazem *nice* (záporná čísla udávají vyšší prioritu - příkaz *nice* bude popisován níže v tomto dílu).
- **SIZE** udává celkovou velikost procesu v paměti (velikost je udávána v kB a je to velikost kódu + velikost zásobníku + velikost dat).

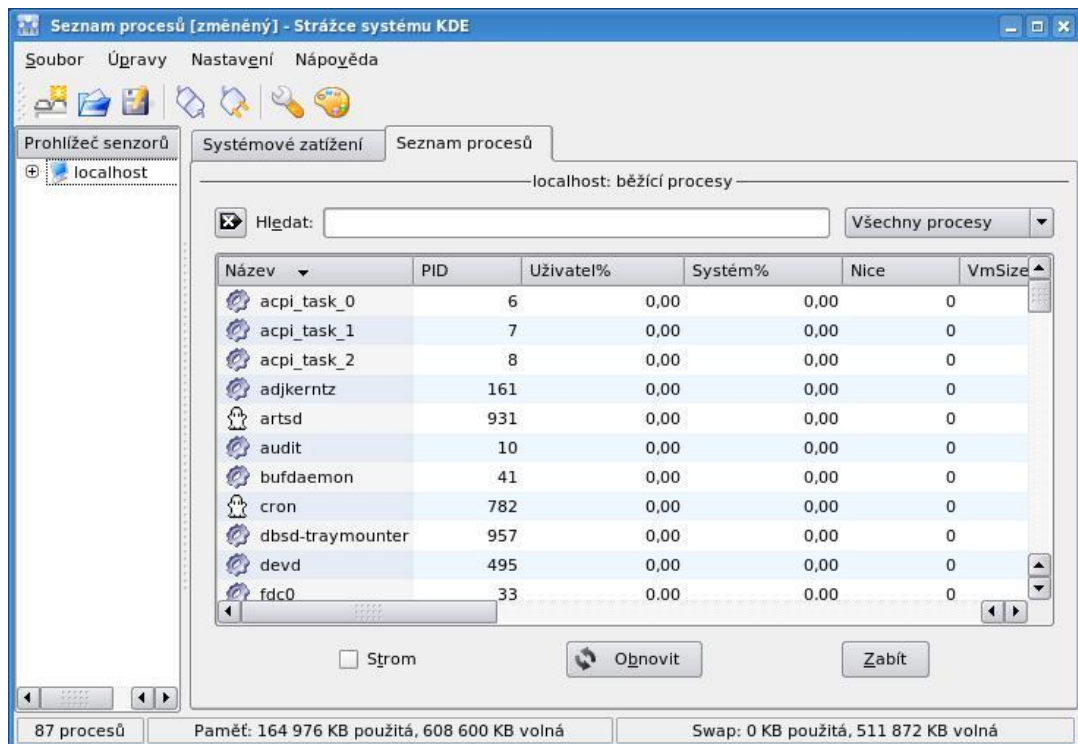
- **RSS** - udává celkovou velikost (v kB) použité fyzické paměti pro daný proces.
- **SHARE** obsahuje velikost sdílené paměti,
- **STAT** udává stav procesu (indikace stavu je stejná jako u příkazu ps výše).
- **%CPU** obsahuje procentuální informace o využití procesoru daným procesem v okamžiku výpisu
- **%MEM** informuje o procentu využití fyzické paměti daným procesem.
- **TIME** udává celkový procesorový čas, po který byl daný proces od spuštění až do okamžiku výpisu zpracováván (z tohoto času je patrné, že spousta procesů má procesorový čas velmi malý, protože mnoho z nich často čeká na nějaké I/O operace).
- **COMMAND** popisuje daný proces/příkaz.

Jak v KDE, tak v Gnome a vlastně i ve kterémkoliv jiném grafickém prostředí je k dispozici grafický správce procesů. Velice často bývá označován jako „System monitor“ neboli „Monitor systému

- V Gnome pod menu „Systém → Administrace → Monitor systému“.
- V KDE: „Menu → Systém → KSysGuard – Monitor výkonu“.



Obrázek 23-3: System monitor v Gnome



Obrázek 23-4: System monitor v KDE

23.3 Stav procesu v Linuxu

- **S** proces usnul - čeká až na něj přijde řada a bude mu přidělen procesor
- **W** paměťový prostor vyhrazený danému procesu byl kompletně uložen na disk (odswapován)
- **R** proces je právě zpracováván procesorem
- **T** proces byl pozastaven
- **D** proces je v nepřerušitelném spánku (v tomto stavu jsou většinou procesy svázané s I/O operacemi)
- **Z** proces, jehož rodičovský proces již ukončil svoji činnost (třeba díky nějaké závažné chybě a nechal po sobě sirotka. Rodičem Zombie procesu se stává Init). Zombie procesy v systému jsou obvykle důsledkem špatně napsaných programů.
- **L** proces má uzamknuté stránky v paměti - platí obvykle pro procesy pracující v reálném čase < proces s vysokou prioritou
- **N** proces s nízkou prioritou

23.4 Priorita procesů v Linuxu

V Linuxu funguje prioritní plánování procesů. Procesy mají přidělenou prioritu při vzniku procesu, v průběhu zpracování procesu se ale může měnit. Priorita procesu, který poslední dobou dostával procesorového času hodně, je OS snížena, naopak je zvýšena priorita takového procesu, který v poslední době dostával procesorového času méně.

Uživatel nemůže přímo ovlivnit hodnotu priority procesu, může pouze nastavit jeden z parametrů, který se při výpočtu priority uplatní. Jedná se o tzv. hodnotu *nice*. Tento parametr má přípustný rozsah od -20 po 19. Výchozí hodnota *nice* je 0. Obyčejný uživatel může hodnotu pouze zvyšovat (a tím snížit prioritu daného procesu), zatímco administrátor může nastavit i zápornou hodnotu *nice* a tím prioritu daného procesu zvýšit.

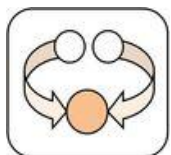
Pro manipulaci s parametrem *nice* slouží:

- Příkazy:
 - *nice* (spuštění aplikace s pozměněnou prioritou) a
 - *renice* (pozměnění priority již běžícího procesu),
- Interaktivní nástroje:
 - *top* pro terminál (příkaz *renice* vyvoláme stiskem "r"),
 - *gtop* pro prostředí GNOME.

Např. následující příkaz zajistí, že aplikace bude spuštěna s nejnižší prioritou:

```
nice -n 19 cesta_k_aplikaci
```

Shrnutí kapitoly



Procesy v systémech unixového typu vznikají pomocí volání jádra `fork()`, které způsobí jakési rozdělení procesu. Proces, který volání jádra `fork()` vyvolal, se nazývá rodičovský proces (parent), pokud `fork()` proběhne úspěšně, vzniká nový proces - tzv. potomek (child). Z toho plyne, že každý proces tedy má svého rodiče vzniká tak hierarchie procesů. V hierarchii funguje jednoduchá komunikace prostřednictvím signálů.

Linux startuje jako process `init`, který spouští služby a tyto služby spouští další procesy.

Každý proces je identifikován číslem PID (Process ID), které procesu přidělí operační systém. Stav procesu a jeho vlastnosti lze monitorovat.

V Linuxu funguje prioritní plánování procesů. Procesy mají přidělenou prioritu při vzniku procesu, v průběhu zpracování procesu se ale může měnit. Uživatel nemůže přímo ovlivnit hodnotu priority procesu, může pouze nastavit jeden z parametrů, který se při výpočtu priority uplatní.

Kontrolní otázky a úkoly



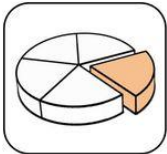
- 1) Jak vznikají procesy a hierarchie procesů v Linuxu?
- 2) Jak se identifikuje proces v Linuxu?
- 3) Jaké informace o procesech lze zjistit?
- 4) Jaký je rozdíl mezi příkazem top a ps?
- 5) Jakou strategii plánování procesoru Linux používá?
- 6) Jak lze ovlivnit prioritu procesů v Linuxu?

Otázky k zamyšlení



- 1) Jaké jsou rozdíly mezi procesy v Linuxu a v OS Windows?

Použitá literatura a jiné zdroje:



- [1] VYCHODIL, Vilém. Operační systém Linux: příručka českého uživatele. 1. vyd. Brno: Computer Press, 2003, 260 s. ISBN 80-722-6333-1.
- [2] MACEK, Petr. Procesy v Linuxu aneb jeden admin vládne všem. *Root.cz* [online]. 11. 1. 2011 [cit. 2011-12-07]. Dostupné z: <http://www.root.cz/clanky/procesy-v-linuxu-aneb-jeden-admin-vladne-vsem/>.
- [3] PODHOLA, Martin. Signály a procesy 1. *LinuxExpres* [online]. 07. prosinec 2005 [cit. 2011-12-07]. Dostupné z: <http://www.linuxexpres.cz/praxe/signaly-a-procesy-1>.