evropský sociální fond v ČR
EVROPSKÁ UNIE
MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY
OP Vzdělávání pro konkurenceschopnost
INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

PSK3-6

Název školy:	Vyšší odborná škola a Střední průmyslová škola, Božetěchova 3
Autor:	Ing. Marek Nožka
Anotace:	Správa procesů v OS Unix/Linux
Vzdělávací oblast:	Informační a komunikační technologie
Předmět:	Počítačové sítě a komunikační technika (PSK)
Tematická oblast:	Operační systém Linux/Unix
Výsledky vzdělávání:	Žák pracuje s procesy a zasílá signály
Klíčová slova:	Linux, Unix, shell, ps, jobs, fg, bg
Druh učebního materiálu:	Online vzdělávací materiál
Typ vzdělávání:	Střední vzdělávání, 4. ročník, technické lyceum
Ověřeno:	VOŠ a SPŠE Olomouc; Třída: 4L
Zdroj:	Vlastní poznámky, Vilém Vychodil: Linux Příručka českého uživatele

Procesy

Proces (program)

Proces (anglicky process) je v informatice název pro spuštěný počítačový program. Proces je umístěn v operační paměti počítače v podobě sledu strojových instrukcí vykonávaných procesorem. Obsahuje nejen kód vykonávaného programu, ale i dynamicky měnící se data, která proces zpracovává. Jeden program může v počítači běžet jako více procesů s různými daty (například vícekrát spuštěný webový prohlížeč zobrazující různé stránky). Správu procesů vykonává operační systém, který zajišťuje jejich oddělený běh, přiděluje jim systémové prostředky počítače a umožňuje uživateli procesy spravovat (spouštět, ukončovat atp.).

V našem výkladu se omezíme pouze na uživatelské hledisko.

Popředí a pozadí

V následujícím textu bude užíváno pojmů "**v popředí**" a "**v pozadí**". Tyto pojmy je nutné chápat z hlediska programů běžících v příkazové

řádce, bez grafického uživatelského rozhraní. Jestliže program běží *v popředí*, neznamená to, že je jeho grafické okno nad ostatními okny, ale že jeho vstup je napojen na terminál.

Jestliže je terminálu (pomocí shell-u) spuštěn program (proces), znamená to, že tento proces je napojen na terminál a v tuto chvíli není možné komunikovat s shell-em.

Spustíme například příkaz `xeyes`. Po dobu běhu tohoto programu v terminálu není možné komunikovat s shell-em a zadávat další příkazy.

```
$ xeyes
fjdkfj dfjdkf
fjkdjfkdjf

jfkdjf

at pisu co pisu, nic se nedeje
```

Klávesové zkratky

Ctrl+C
: Proces, který je právě *v popředí* bude ukončen (přerušen). (Bude mu zaslán signál SIGINT)

Ctrl+Z
: Proces, který je právě *v popředí* bude pozastaven (suspended). (Bude mu zaslán signál SIGSTOP)

Práce s procesy

Nyní po spuštění programu `xeyes` stiskneme `Ctrl+Z` a oči uspíme:

```
$ xeyes
^Z
[1]+  Pozastavena             xeyes
$
```

V tuto chvíli je terminál volný pro spuštění dalších příkazů. Hned toho využijeme a podíváme se kolik je hodin (program `xclock`). Program ale opět pozastavíme pomocí `Ctrl+Z`.

```
$ xclock -update 1
^Z
[2]+  Pozastavena             xclock -update 1
$
```

Pozastavený proce spí. Je zmražen. Vteřinová rafička hodin se nehýbe a oči nesledují kurzor myši. Do třetice spustíme a pozastavíme program `xman`. (Slouží pro čtení zobrazení manuálových stránek)

```
$ xman
^Z
[3]+  Pozastavena             xman
```

```
$
```

Seznam procesů běžících v aktuálním shell-u vydá příkaz **jobs**:

```
$ jobs
[1]   Pozastavena             xeyes
[2]-  Pozastavena             xclock -update 1
[3]+  Pozastavena             xman
```

Příkaz **fg** (foreground) slouží pro přesun programu do popředí. Program se tedy znovu napojí na terminál a je možné s ním komunikovat nebo ukončit ho (`Ctrl+C`) nebo znovu pozastavit (`Ctrl+Z`). Jako parametr udáváme `%N`, kde `N` je číslo procesu, které je v levém sloupci ve výpisu `jobs`.

```
$ fg %2
xclock -update 1
```

Můžeme pozorovat, že vteřinová ručička hodin se opět rozběhne.

Příkaz **bg** (background) slouží pro přesun programu na pozadí. To znamená, že proces poběží, ale nebude napojen na terminál a bude proto možné používat shell nebo spustit jiný program.

```
$ fg %2
xclock -update 1
^Z
[2]+  Pozastavena             xclock -update 1
$ bg %2
[2]+ xclock -update 1 &
$
```

Je třeba dodat, že je nesmysl dávat na pozadí programy, které s uživatelem interaktivně komunikují. Odstrašujícím, příkladem může celoobrazovkový správce souborů Midnight Commander -- `mc`.

Pomocí metaznaku `&`, který zapíšeme za příkaz, můžeme umístit program (proces) na pozadí rovnou při jeho spuštění:

```
$ oclock &
[4] 7321
$ jobs
[1]-  Pozastavena             xeyes
[2]   Běží                  xclock -update 1 &
[3]+  Pozastavena             xman
[4]   Běží                  oclock &
$
```

Příkaz `fg` a `bg` umožňuje vynechat parametr s číslem procesu. Potom je na popředí nebo pozadí umístěn program, se kterým se naposledy pracovalo a který má v levém sloupci ve výpisu `jobs` znak `+`.

```
$ jobs
[1]-  Pozastavena             xeyes
```

```
[2]   Běží                    xclock -update 1 &
[3]+  Pozastavena               xman
[4]   Běží                    oclock &
$ fg
xman
```

Procesy (ne)napojené na (jiný) terminál

Výše popsané, lze uplatnit pouze na procesy, které běží v aktuálním shell-u. Příkaz `jobs` jiné procesy, které běží v jiném shell-u nebo terminálu "nevidí". Pokud požadujeme víc použijeme příkaz `ps`

```
$ ps
  PID TTY          TIME CMD
 4325 pts/5    00:00:00 bash
 4370 pts/5    00:00:00 xeyes
 5258 pts/5    00:00:00 xclock
 7321 pts/5    00:00:00 oclock
 8466 pts/5    00:00:00 ps
15276 pts/5    00:00:01 zsh
```

Číslo v levém sloupci je tzv. *Process identifier* `PID`. Je to číslo, které proces jednoznačně identifikuje. Pokud je příkaz `ps` použit bez parametrů vypisuje opět jen procesy běžící v aktuálním shell-u. Pokud přidáme parametr `-a` budou vypsány všechny procesy běžící napojené na jakýkoliv terminál.

```
$ ps -a
  PID TTY          TIME CMD
 4325 pts/5    00:00:00 bash
 4370 pts/5    00:00:00 xeyes
 5258 pts/5    00:00:00 xclock
 5350 tty2     00:00:00 zsh
 5360 tty2     00:00:00 startx
 5605 tty2     00:00:04 urxvt
 5631 pts/9    00:00:00 mc
 5671 pts/7    00:00:00 su
 5681 pts/7    00:00:07 zsh
 5802 tty2     00:16:36 psi-plus
 7321 pts/5    00:00:00 oclock
 8709 pts/5    00:00:00 ps
 8710 pts/5    00:00:00 bash
16616 tty2     00:00:00 urxvt
17205 tty2     00:00:00 py.white
17210 tty2     00:00:42 ipython
18443 tty2     00:00:00 sh
18444 tty2     00:00:00 urxvt
21095 tty2     00:04:18 claws-mail
24985 pts/7    00:00:22 aptitude
30285 tty2     00:28:07 iceweasel
```

Program `ps` má velké množství voleb, jejichž popis najdeme v manuálové stránce. Zde uvedeme několik příkladů:

příklad	**popis**
`ps ux`	vypíše všechny procesy aktuálního uživatele
`ps fx`	
`-- f...`	vypíše všechny procesy aktuálního uživatele včetně

`ps Tux`	stromu procesů
`ps ax`	vypíše všechny procesy všech uživatelů
`ps uxw`	
`ps uxww`	
`ps uxwww`	vypíše všechny procesy aktuálního uživatele ve tvaru jak byly zadány na příkazový řádek (každé `w` prozradí něco víc)

Zasílání signálů

Pro zasílání signálů slouží program `kill`.

Uživatelsky nejdůležitější signály

signál	**význam**
`SIGTERM`	nenásilné ukončení procesu
`SIGKILL`	okamžité ukončení procesu
`SIGSTOP`	pozastavení procesu
`SIGCONT`	probuzení -- pokračování procesu

Situaci budeme ilustrovat na následujícím příkladu: Před hodinou uživatel `karel` spustil náročnou úlohu pro výpočet čísla π. Výpočet by ale nyní rád pozastavil, protože spotřebovává všechen strojový čas a `karel` potřebuje na počítači chvíli pracovat.

Nejprve zjistí `PID` programu:

```
$ ps axwww | egrep mojePI
 5463 tty2     TN   48:41 ./mojePI
10256 pts/12   S+    0:00 egrep mojePI
```

... a zašle signál pro pozastavení: Program `kill` přebírá jako parametr `PID` procesu. (Nebo `%N` stejně jako `fg` a `bg`.)

```
$ kill -SIGSTOP 5463
```

V tuto chvíli proces usnul a strojový čas je k dispozici jiným úlohám. Pro pokračování výpočtu potom slouží příkaz

```
$ kill -SIGCONT 5463
```

Pro ukončení

```
$ kill -SIGTERM 5463
$ kill -SIGKILL 5463
```

Interaktivní správa procesů

Pro interaktivní správu procesů slouží program `top` a jeho barevnější a vylepšená varianta `htop`.

- ps
- kill
- top
- htop
- nice
- killall
- pkill
- pgrep