Ćwiczenie 2

Cel ćwiczenia

Celem ćwiczenia poznanie poleceń przeznaczonych do zarządzania procesami w systemie operacyjnym Linux.

Proszę zapoznać się z praktycznym działaniem każdego z wyszczególnionych tutaj poleceń. Użycie każdego polecenia należy udokumentować w sprawozdaniu, które powinno zawierać nazwę wydawanego polecenia i jego krótki opis, rezultaty działania polecenia oraz opis otrzymanych rezultatów.
Przy tworzeniu sprawozdania wspomóc się można plikiem logu rejestrującego działanie poszczególnych poleceń. Jednakże, należy pamiętać, aby nie tworzyć zbyt dużych logów bowiem trudno będzie je przeglądać.
Sprawozdanie w postaci pliku PDF proszę przesłać na adres e-mailowy: jpszub@matman.uwm.edu.pl podając w tytule e-maila: „ASK Lab2 i swoje nazwisko".

Etapy startu systemu

Na współczesnych platformach x86 procedura startu Linuxa wygląda to następująco[1]:

1. Uruchamiany jest firmware UEFI (Unified Extensible Firmware Interface) /BIOS
   - UEFI stanowi interfejs pomiędzy systemem operacyjnym a firmwarem spełnia tre same funkcje co BIOS, lecz pozbawiany jest jego ograniczeń;
   - UEFI działając w trybie wirtualnego adresowania pamięci, umożliwia bezpośrednie załadowanie jądra systemu do pamięci z pominięciem bootloader'a. Pomimo tego, dystrybucje Linuxa dostarczają program rozruchowy;
   - UEFI zamiast MBR korzysta z partycji (EFI System Partition) ESP i z niej ładuje program rozruchowy lub jądro systemu, umożliwiając bezpośrednie ładowanie programów o rozmiarze wielu MB;
   - ESP zawiera wiele programów rozruchowych, których wybór dokonywany jest z menu UEFI podczas uruchamiania komputera;
2. Ładowany jest program rozruchowy (bootloader), który współcześnie stał się de facto opcją;
   - większość dystrybucji Linuxa korzysta z bootloadera GRUB 2;
3. Ładowane jest jądro Linuxa.
   - Jądro przechowywane w postaci spakowanego obrazu `bzImage` zostaje rozpakowane do postaci binarnej i inicjalizuje tablice stron;
   - Wywoływana jest funkcja `start_kernel();`
   - Montowany jest inicjalny system plików `initramfs`;
4. Uruchamiany jest proces `init`;
   - Wybiera „run level" wykonując odpowiadający mu skrypt startowy `rc`;
5. Montowany jest `rootfs`;
6. Przygotowanie przestrzeni użytkownika;
   - `systemd` inicjalizuje połączenia sieciowe, montuje wirtualne wolumeny, uruchamia środowisko graficzne, wyświetla ekran logowania oraz startuje usługi takie jak serwer ssh, http, mysql, czy serwis uruchamiający maszyny wirtualne.

Poziomy pracy systemu (run levels)[2]

Niektóre poziomy pracy są standardem systemu operacyjnego Linux, a inne zależą od jego dystrybucji.
Następujące poziomy pracy są standardowe:
0 – Postój (Wyłącza system;
1 – Tryb pojedynczego użytkownika
- System uruchamia się w trybie administratora bez uruchamiania demonów i sieci.

6 – Restart
Poziomy działania w zakresie 2 – 5 różnią się w zależności od dystrybucji:
- W Ubuntu i Debianie poziomy pracy zapewniają pełny tryb dla wielu użytkowników z obsługą sieci i graficznym loginem.
- W Fedorze i Red Hat, runlevel 2 zapewnia tryb wielu użytkowników bez sieci (tylko logowanie do konsoli), runlevel 3 zapewnia tryb wielu użytkowników z obsługą sieci (tylko logowanie do konsoli), poziom uruchamiania 4 jest nieużywany, a poziom działania 5 zapewnia tryb wielu użytkowników, logowanie sieciowe i środowisko graficzne.

Poziom uruchamiania, można wybrać za pomocą boot loadera – Grub. W tym celu na początku procesu uruchamiania należy nacisnąć klawisz, aby uzyskać dostęp do Gruba, wybrać wpis rozruchowy i go edytować. W celu uruchomienie opcji należy wybrać Ctrl + x.

Przełączanie na inny poziom pracy

Aby przełączyć się na inny poziom uruchamiania, gdy system jest już uruchomiony, należy skorzystać z polecenia:

```
sudo telinit #
```

gdzie znak # należy zastąpić numerem poziomu działania, na który system ma zostać przełączony. Jeżeli dystrybucja nie zawiera `sudo` polecenie należy uruchomić jako `root`.

Zadanie 1

Sprawdź aktualny poziomy pracy systemu. Udokumentuj swoje działania za pomocą odpowiednich zrzutów ekranu.

Polecenie `initctl list`

Umożliwia administratorowi systemu komunikację z konfiguracją demona `init` podając listę usług uruchamianych wraz z systemem.

Polecenie `service`

Głównym celem tego polecenia jest uruchamianie i zatrzymywanie skryptów oraz tworzenie procesów. Można się nim posłużyć do sprawdzania statusów usług w systemie. Listę usług pokazuje polecenie:

```
service -- status-all
```

Zadanie 2

Sprawdź jakie usługi zostały uruchomione w systemie. Wykonaj odpowiedni zrzut ekranu.

Zamykanie systemu (Shutting the System Down)

Etapy:

- Wylogowanie użytkownika;
- Zatrzymanie usług;
- Zapisanie danych;
- Odmontowanie systemu plików;
- Wyłączenie zasilania systemu;

Odmiany zamykania systemu (Types of shutdowns):

- `init 0` – bez ostrzeżenia użytkownika `init 0` zmienia bieżący poziom pracy systemu na `run level 0`. Ten mod może być uruchamiany jedynie przez superusera;
- `shutdown -h time "message"`- np. ostrzega użytkownika o zamknięciu systemu z wyprzedzeniem 5 minut:

```
# shutdown -h +5 "Development server is going down for
maintenance. Please save your work ASAP."
```

Polecenia zamknięcia system:

`halt` i `reboot` –

Powiązane polecenia i pliki

Polecenia	Pliki
`init` - parent of all processes	`/etc/inittab`
`runlevel` - tell you which runlevel	`/etc/rc.d/*`
`reboot` and `ctrl-alt-del`- reboots the system	`/etc/rc.d/init.d`
`halt` and `shutdown`- powers down the system	`/etc/rc.d/rc*.d`

Zarządzanie procesami[3]

Proces jest elementarną jednostką aktywności zarządzaną przez system operacyjny, która ubiega się o zasoby systemu komputerowego w celu wykonania programu. Proces w systemie UNIX można więc rozumieć jako wykonywany w systemie program.[4]
W skład procesu wchodzą następujące elementy:

- kod binarny — definiuje zachowanie procesu,
- dane programu — zbiór wartości przetwarzanych oraz wyniki,
- zasoby tworzące środowisko wykonawcze, np. pamięć CPU itp.,
- blok kontrolny procesu (PCB, deskryptor) — opis bieżącego stanu procesu.

Deskryptor procesu (blok kontrolny procesu, PCB) —używany przez zarządcę procesów w celu rejestrowania stanu procesu w czasie jego monitorowania. W jego skład wchodzą:

- identyfikator procesu,
- stan procesu (nowy, gotowy, oczekujący, itd.),
  - pracujący w trybie użytkownika (proces znajduje się na procesorze i wykonuje kod),
  - pracujący w trybie jądra (jądro wykonuje wywołanie systemowe, wykonane przez proces)

    - uśpiony (proces czeka na jakieś zdarzenie, np. na odczyt danych z dysku lub otrzymanie danych z sieci)
    - gotowy do wykonania (może być uruchomiony w każdej chwili, jednak nie ma jeszcze przydzielonego procesora)
    - zombie (proces zakończył działanie i czeka na odebranie kodu powrotu przez proces macierzysty),
- identyfikator właściciela,
- identyfikator przodka,
- lista przydzielonych zasobów,
- zawartość rejestrów procesora,
- prawa dostępu (domena ochrony),
- informacje na potrzeby zarządzania pamięcią,
- informacje na potrzeby planowania (np. priorytet),
- informacje do rozliczeń,
- wskaźniki do kolejek.

Deskryptor zasobu — przechowuje informacje o dostępności i zajętości danego typu zasobu.

Polecenie `ps`

`ps` generuje listę charakteryzującą wszystkie aktywne procesy w systemie.[5]

```
[root@localhost ~]# ps -eo euser, ruser, fuser, comm
ERROR: Unsupported option (BSD syntax)
********* simple selection *********  ********* selection by list *********
-A all processes                      -C by command name
-N negate selection                   -G by real group ID (supports names)
-a all w/ tty except session leaders  -U by real user ID (supports names)
-d all except session leaders         -g by session OR by effective group name
-e all processes                      -p by process ID
T  all processes on this terminal     -s processes in the sessions given
a  all w/ tty, including other users  -t by tty
g  OBSOLETE -- DO NOT USE             -u by effective user ID (supports names)
r  only running processes             U  processes for specified users
x  processes w/o controlling ttys     t  by tty
*********** output format **********  *********** long options ***********
-o,o user-defined  -f full            --Group --User --pid --cols --ppid
-j,j job control   s  signal          --group --user --sid --rows --info
-O,O preloaded -o  v  virtual memory  --cumulative --format --deselect
-l,l long          u  user-oriented   --sort --tty --forest --version
-F   extra full    X  registers       --heading --no-heading --context
                  ********* misc options *********
-V,V  show version      L  list format codes  f  ASCII art forest
-m,m,-L,-T,H  threads   S  children in sum    -y change -l format
-M,Z  security data     c  true command name  -c scheduling class
-w,w  wide output       n  numeric WCHAN,UID  -H process hierarchy
[root@localhost ~]# _
```

Opis opcji polecenia `ps`:

- `-a` - wyświetla wszystkie procesy posiadające terminal kontrolny, nie tylko bieżące procesy użytkowników,
- `-r` - wyświetla tylko pracujące procesy,
- `-x` - wyświetla procesy nie posiadające terminala kontrolnego,
- `-u` - wyświetla właścicieli procesów,
- `-f` - wyświetla powiązania między procesami - format drzewiasty,
- `-l` - generuje listę w długim formacie,
- `-w` - wyświetla parametry linii poleceń procesu (do polowy linii),
- `-ww` - wyświetla wszystkie parametry linii poleceń procesu niezależnie od jej długości.
- `-j` – identyfikatory procesów: PGID[6], SID,

-s - format sygnału,
-m - wyświetla informacje o pamięci.
-S - dodaje czas cpu potomka i błędy stron.
-h - bez nagłówka.
-n - wyjście numeryczne dla USER i WCHAN.

Przykład `ps -l`

```
[root@localhost etc]# ps -l
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S     0   870   848  0  80   0 -  1363 -      tty1     00:00:00 bash
4 R     0  1110   870  0  80   0 -  1217 -      tty1     00:00:00 ps
[root@localhost etc]# _
```

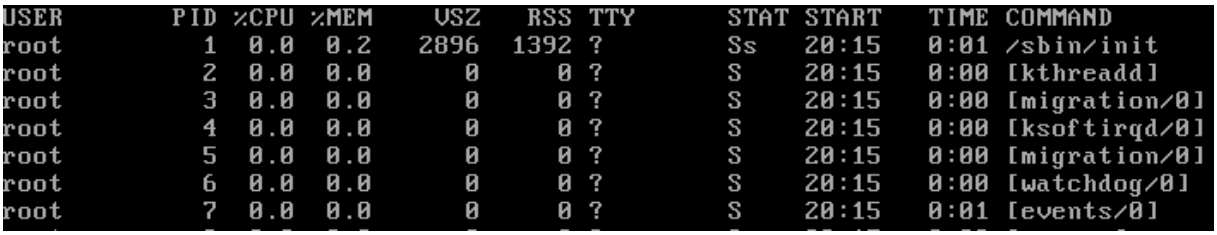
Ekran przedstawia następujące informacje:

- `F` - FLAG: process startuje bez uprawnień superusera,
- `4` proces z uprawnieniami superusera. Check **man ps** for more info,
- `S` – STATE proces aktualnie działający,
- `UID` - User ID identyfikator użytkownika, który zainicjował proces. UID is actual an alias for `EUID` (Effective User ID).
- `PID` - Process ID.
- `PPID` - Parent Process ID. Jest to ID procesu rodzicielskiego danego procesu (from which proces had been forked from).
- `C` - Integer value przedstawiająca wykorzystania procesora w %.
- `PRI` – Priorytet procesu. Im większa liczba tym priorytet mniejszy.
- `NI` - Nice wartość z zakresu -20 to 19. Im większa wartość tym bardziej użytkownik jest "milszy" dla pozostałych użytkowników. Innymi słowy im większa wartość tym mniejszy priorytet.
- `Sz` – Wykorzystanie pamięci wirtualnej.
- `WCHAN` – Adres pamięci zdarzenia na, na które proces oczekuje.
- `TT` lub `TTY` – Terminal powiązany z procesem.
- `TIME` Total CPU usage.

Polecenie wyświetlające wszystkie uruchomione procesy:
`ps aux | less ps -auxww.`
Gdzie:

- `-a`: select all processes on a terminal, including those of other users,
- `-u`: select by user ID
- `-x`: select processes without ttys,
- `-l` (long) additional info about each process:

Ten zestaw opcji wyświetla wszystkie procesy (bez względu na to, czy kontrolują one terminal), właściciela każdego procesu oraz wszystkie parametry procesu:



```
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.2   2896  1392 ?        Ss   20:15   0:01 /sbin/init
root         2  0.0  0.0      0     0 ?        S    20:15   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S    20:15   0:00 [migration/0]
root         4  0.0  0.0      0     0 ?        S    20:15   0:00 [ksoftirqd/0]
root         5  0.0  0.0      0     0 ?        S    20:15   0:00 [migration/0]
root         6  0.0  0.0      0     0 ?        S    20:15   0:00 [watchdog/0]
root         7  0.0  0.0      0     0 ?        S    20:15   0:01 [events/0]
```

W pierwszym wierszu znajduje się opis poszczególnych kolumn raportu:
`USER` - kto jest właścicielem danego procesu.
`PID` - numer identyfikacyjny procesu.
`%CPU` - procentowe wykorzystanie procesora.
`%MEM` - procentowe wykorzystanie pamięci przez proces.
`VSZ` - wielkość pamięci wirtualnej przydzielonej procesowi.
`RSS` - wielkość rzeczywistej pamięci wykorzystywanej przez proces.
`TTY` - terminal kontrolny procesu. Znak `?` w tej kolumnie oznacza, że proces nie jest połączony z żadnym terminalem kontrolnym.
`STAT` - stan procesu:

- `S` - proces uśpiony, proces gotowy do uruchomienia, lecz procesor jest zajęty.
- `R` - proces jest aktualnie wykonywany.
- `D` - proces jest uśpiony, oczekuje na operacje (zazwyczaj I/O).
- `T` - proces jest debuggowany lub został zatrzymany.
- `Z` – zombie, co oznacza, że albo proces macierzysty nie potwierdził śmierci procesu potomnego lub proces macierzysty został niewłaściwie zabity i dopóki nie zostanie kompletnie zabity, proces `init` nie będzie mógł korzystać z tego procesu.

`START` - czas lub data rozpoczęcia procesu.
`TIME` - przedział czasu wykorzystany przez CPU.
`COMMAND` - nazwa procesu oraz jego parametry.

Priorytety:

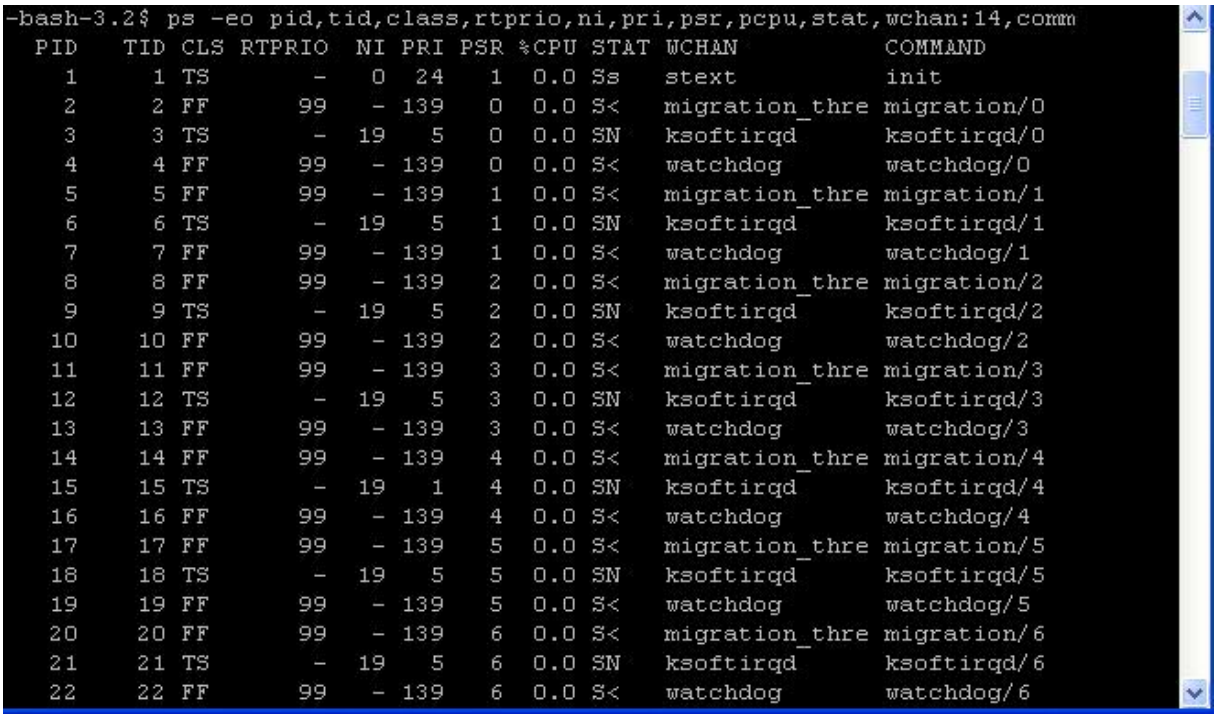
Komenda ustala priorytet procesu w algorytmie szeregowania, zgodnie z zasadą, że procesowi o większym priorytecie zostanie przyznane więcej czasu CPU niż procesowi o mniejszym priorytecie.

Priorytet określa się parametrem zwanym `niceness`, będącym liczbą całkowitą z przedziału `n=-20` (najwyższy priorytet) do `19` (najniższy priorytet). Tylko użytkownik o uprawnieniach superusera może używać ujemnych wartości parametru `niceness`.

W Linuxie poprzez edycję pliku `/etc/security/limits.conf`, można to umożliwić innym użytkownikom i grupom. Procesy uruchamiane bez użycia komendy `nice` mają domyślnie priorytetu równy 0, ale administrator systemu może to zmienić.

```
nice

nice -n „PID procesu”
```



```
-bash-3.2$ ps -eo pid,tid,class,rtprio,ni,pri,psr,pcpu,stat,wchan:14,comm
  PID   TID CLS RTPRIO  NI PRI PSR %CPU STAT WCHAN          COMMAND
    1     1 TS       -   0  24   1  0.0 Ss   stext          init
    2     2 FF      99   - 139   0  0.0 S<   migration_thre migration/0
    3     3 TS       -  19   5   0  0.0 SN   ksoftirqd      ksoftirqd/0
    4     4 FF      99   - 139   0  0.0 S<   watchdog       watchdog/0
    5     5 FF      99   - 139   1  0.0 S<   migration_thre migration/1
    6     6 TS       -  19   5   1  0.0 SN   ksoftirqd      ksoftirqd/1
    7     7 FF      99   - 139   1  0.0 S<   watchdog       watchdog/1
    8     8 FF      99   - 139   2  0.0 S<   migration_thre migration/2
    9     9 TS       -  19   5   2  0.0 SN   ksoftirqd      ksoftirqd/2
   10    10 FF      99   - 139   2  0.0 S<   watchdog       watchdog/2
   11    11 FF      99   - 139   3  0.0 S<   migration_thre migration/3
   12    12 TS       -  19   5   3  0.0 SN   ksoftirqd      ksoftirqd/3
   13    13 FF      99   - 139   3  0.0 S<   watchdog       watchdog/3
   14    14 FF      99   - 139   4  0.0 S<   migration_thre migration/4
   15    15 TS       -  19   1   4  0.0 SN   ksoftirqd      ksoftirqd/4
   16    16 FF      99   - 139   4  0.0 S<   watchdog       watchdog/4
   17    17 FF      99   - 139   5  0.0 S<   migration_thre migration/5
   18    18 TS       -  19   5   5  0.0 SN   ksoftirqd      ksoftirqd/5
   19    19 FF      99   - 139   5  0.0 S<   watchdog       watchdog/5
   20    20 FF      99   - 139   6  0.0 S<   migration_thre migration/6
   21    21 TS       -  19   5   6  0.0 SN   ksoftirqd      ksoftirqd/6
   22    22 FF      99   - 139   6  0.0 S<   watchdog       watchdog/6
```

W pierwszym wierszu znajduje się opis poszczególnych kolumn raportu:
PID – identyfikator procesu;
TID – identyfikator wątku;
CLS – klasa schedulera procesu (scheduling class of the process) możliwe są następujące wartości: [7]

- – nieraportowany,
- TS SCHED_OTHER – dokonuje kolejkowania karuzelowego, w którym kolejkowanie określonego zadania zależy od innych zadań uruchomionych w systemie.
- FF SCHED_FIFO – kolejkuje procesy algorytmem FIFO,
- RR SCHED_RR – dokonuje kolejkowania karuzelowego (round-robin) w określonym wycinku czasu,
- B SCHED_BATCH – stosuje się do długotrwałych i nieinteraktywnych procesów, które aby zapewnić interaktywność są często przerywane przez inne procesy,
- ? wartość nieznana;

RTPRIO – priorytet czasu rzeczywistego.
NI – parametr określający preferencje użytkownika dotyczące priorytetu danego procesu 0 ustawianego poleceniem nice wartości N=(-20…+19), początkowa wartość NI =0.
PRI – priorytet procesu.
PSR – procesor, do którego proces jest obecnie przypisany.
%CPU – procentowe wykorzystanie procesora.
STAT - stan procesu:

- S - proces uśpiony, proces gotowy do uruchomienia, lecz procesor jest zajęty.
- R - proces jest aktualnie wykonywany.
- D - proces jest uśpiony, oczekujący na operacje (zazwyczaj I/O).
- T - proces jest debuggowany lub został zatrzymany.

- `Z` – zombie, co oznacza, że albo proces macierzysty nie potwierdził śmierci procesu potomnego lub proces macierzysty został niewłaściwie zabity i dopóki nie zostanie kompletnie zabity, proces `init` nie będzie mógł korzystać z tego procesu.

`WCHAN` – Adres funkcji jądra, w której proces został uśpiony, zadania aktulanie uruchomione będą miały w tej kolumnie wyświetlony myślnik '`-`'.
`CMD` – polecenie (wraz z parametrami), które spowodowało uruchomienie procesu.

Szczegółowy opis polecenia `ps` można znaleźć na następujących stronach WWW[8]

Zadanie 3

Pokaż środowisko po wydaniu polecenia `# ps –…`
gdzie:

- `-A`: wybiera wszystkie procesy,
- `-e`: wyświetla pełną listę procesów,
- `-f`: (full) wyświetla dodatkowe informacje o procesach.

Zadanie 4

Pokaż wszystkie procesy, których właścicielem jest `root`, a następnie procesy z wyjątkiem tych, których właścicielem jest `root`.

Hierarchię procesów można obejrzeć za pomocą polecenia:
`ps [opcje] --forest`
lub korzystając z opcji `-f` polecenia `ps`.
`# ps -[opcje]f | more`

Informacje o wątkach (threads)

Polecenie
`# ps -Elf`

```
UID          PID  PPID   LWP  C NLWP STIME TTY          TIME CMD
root           1     0     1  0    1 20:15 ?        00:00:01 /sbin/init
root           2     0     2  0    1 20:15 ?        00:00:00 [kthreadd]
root           3     2     3  0    1 20:15 ?        00:00:00 [migration/0]
root           4     2     4  0    1 20:15 ?        00:00:00 [ksoftirqd/0]
root           5     2     5  0    1 20:15 ?        00:00:00 [migration/0]
root           6     2     6  0    1 20:15 ?        00:00:00 [watchdog/0]
root           7     2     7  0    1 20:15 ?        00:00:03 [events/0]
root           8     2     8  0    1 20:15 ?        00:00:00 [cgroup]
root           9     2     9  0    1 20:15 ?        00:00:00 [khelper]
root          10     2    10  0    1 20:15 ?        00:00:00 [netns]
root          11     2    11  0    1 20:15 ?        00:00:00 [async/mgr]
```

`LWP` - Shows the lightweight process ID.
`NLWP` - The number of threads in the processes.

Informacje dotyczące użytkownika systemu

Procesy danego użytkownika:
`# ps -eo user`
lub
`# ps -To user`

Zadanie 5

Sprawdź kto jest użytkownikiem procesów w twoim systemie.

Informacje dotyczące bezpieczeństwa systemu

Opcje LABEL podające informacje o bezpieczeństwie systemu.

```
# ps -aZ
```

```
[root@localhost ~]# ps -aZ
LABEL                              PID TTY          TIME CMD
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 950 tty1 00:00:00 ps
[root@localhost ~]# _
```

W systemach Security-Enhanced Linux (SEL Linux) wszystkie procesy i pliki są oznaczone w sposób reprezentujący informacje istotne z punktu widzenia bezpieczeństwa.
Polityka bezpieczeństwa oparta jest na koncepcji zapewnienia jak najmniejszej liczby uprawnień dla danego obiektu (np. konta użytkownika, urządzenia, programu) niezbędnych dla jego poprawnego funkcjonowania. Ogranicza to możliwości wyrządzenia szkód po przejęciu kontroli nad obiektem przez osoby niepowołane.
Polityka Targeted, w sposób domyślny, sprawia, że jedynie procesy objęte politykami są chronione w ograniczonej domenie (`confined domain`), pozostałe działaja w domenie `unconfined`.
Domyślnie są to procesy użytkowników, które funkcjonują tak, jakby nie było SELinux.[9]

Program `top`

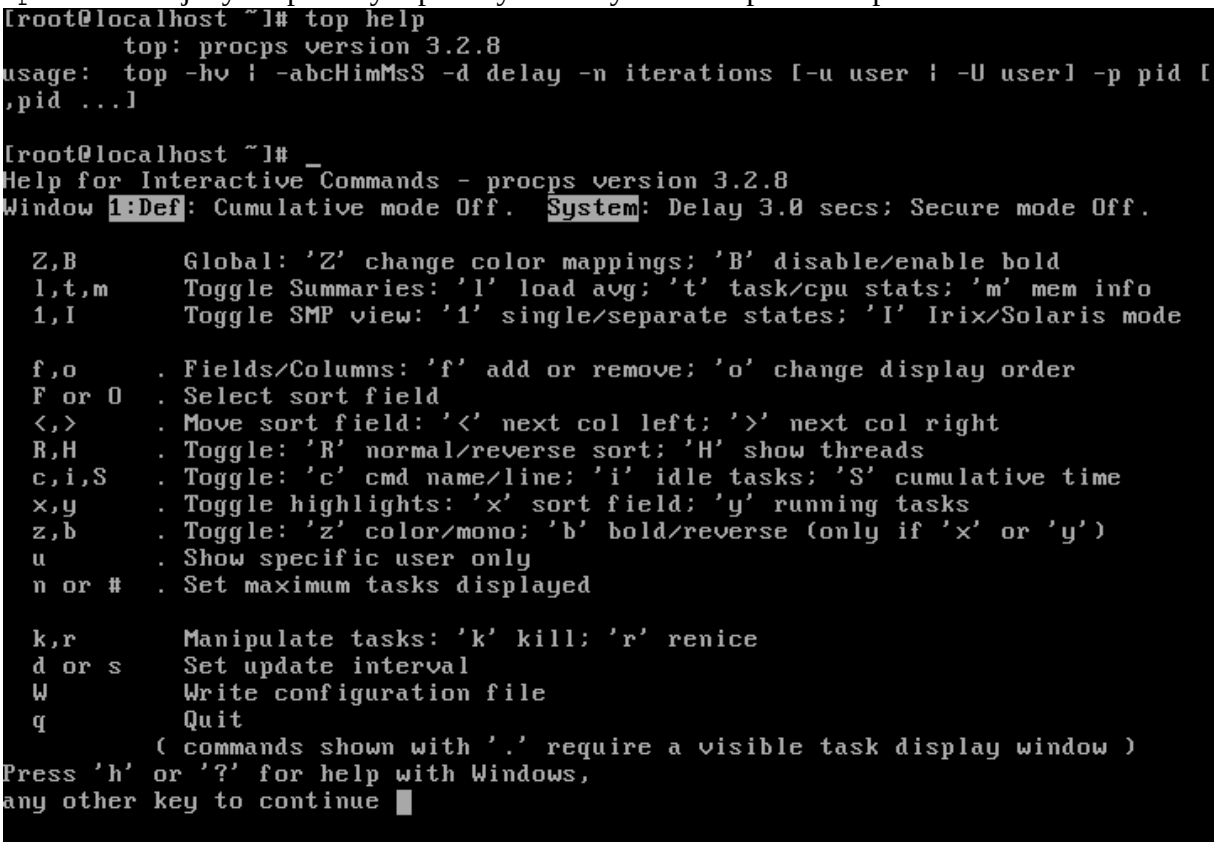
Polecenie `top` zapewnia z poziomu konsoli dynamiczny podgląd bieżącego stanu systemu w czasie rzeczywistym. Jest to interaktywna wersja polecenia `ps`. Zamiast wyświetlać statyczną listę procesów w systemie, `top` domyślnie odświeża listę co 2,5s co stanowi obciążenie CPU. Ten przedział czasu można odpowiednio zmienić. Wywołajmy polecenie `top` bez żadnych parametrów:

```
top
```

```
top - 22:43:11 up  2:28,  1 user,  load average: 0.00, 0.00, 0.00
Tasks:  61 total,   1 running,  60 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.0%us,  0.3%sy,  0.0%ni, 99.7%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:    510824k total,   144400k used,   366424k free,     8224k buffers
Swap:   835580k total,        0k used,   835580k free,    90016k cached
_
  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 1142 root      20   0  2564 1096  896 R  0.7  0.2   0:00.16 top
    1 root      20   0  2896 1392 1196 S  0.0  0.3   0:01.12 init
    2 root      20   0     0    0    0 S  0.0  0.0   0:00.01 kthreadd
    3 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 migration/0
    4 root      20   0     0    0    0 S  0.0  0.0   0:00.03 ksoftirqd/0
    5 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 migration/0
    6 root      RT   0     0    0    0 S  0.0  0.0   0:00.55 watchdog/0
    7 root      20   0     0    0    0 S  0.0  0.0   0:03.28 events/0
    8 root      20   0     0    0    0 S  0.0  0.0   0:00.00 cgroup
    9 root      20   0     0    0    0 S  0.0  0.0   0:00.00 khelper
   10 root      20   0     0    0    0 S  0.0  0.0   0:00.00 netns
   11 root      20   0     0    0    0 S  0.0  0.0   0:00.00 async/mgr
   12 root      20   0     0    0    0 S  0.0  0.0   0:00.00 pm
   13 root      20   0     0    0    0 S  0.0  0.0   0:00.08 sync_supers
   14 root      20   0     0    0    0 S  0.0  0.0   0:00.08 bdi-default
   15 root      20   0     0    0    0 S  0.0  0.0   0:00.00 kintegrityd/0
   16 root      20   0     0    0    0 S  0.0  0.0   0:00.16 kblockd/0
   17 root      20   0     0    0    0 S  0.0  0.0   0:00.00 kacpid
```

Pomoc dotycząca `top`:
`-d` – parameter definiuje odstęp czasu pomiedzy kolejnymi uaktualnieniami ekranu.

`-p` - monitoruje tylko procesy o podanym identyfikatorze procesu itp.



Monitorować działania wybranych aplikacji można osiągnąć poprzez wykorzystanie polecenia:

```
top -p [PID_procesu1, PID_procesu1,…]
```

Po wywołaniu polecenia zostaną wyświetlone tylko informacje odnoszące się do zdefiniowanych numerów PID procesów.

Zadanie 6

Zbadaj za pomocą polecenia `top` działanie kilku wybranych procesów.

Aby włączyć monitorowanie procesów wybranego użytkownika należy skorzystać z polecenia:

```
top -u [nazwa użytkownika]
```

Zadanie 7

Zbadaj za pomocą polecenia `top` działanie procesów uruchomionych przez wybranego użytkownika.

W ostatnich zadaniach może okazać się przydatne przekierowania wyników działania `top` do pliku:

```
# top -b -n1 > /tmp/process.log
```

Można także wyniki przesłać pocztą elektroniczną:

```
# top -b -n1 | mail -s 'Process snapshot' you@example.com
```

Zatrzymanie programu:

```
top q
```

Inne narzędzia pozwalające obejrzeć hierarchię procesów: `htop` i `atop`

`htop` podobnie jak `top` jest interaktywną przeglądarką procesów, która pozwala dodatkowo na skrolowania pionowe I poziome okna umożliwiając obejrzenie informacji na temat wszystkich procesów.
Zadania powiązane z procesami jak (killing, renicing) można wykonać bez wprowadzania identyfikatorów PID.

W celu zainstalowania `htop` należy wpisać polecenie:

```
# apt-get install htop
```

lub

```
# yum install htop
```

A następnie uruchomić wpisując w linii poleceń:

```
# htop
```

Wyjście z programu `q`, uzyskanie pomocy `h`.

Drzewo procesów

W Linuxie każdy proces w systemie ma swój proces macierzysty. Polecenie `psstree` pokazuje strukturę uruchomionych procesów w formie drzewa. Drzewo znajduje się albo w `pid` albo w `init` jeśli `pid` pominięto. Jeśli podano nazwę użytkownika, wszystkie drzewa procesów zakorzenione w procesach przez tego użytkownika są widoczne.

Polecenie `pstree`

Procesy tworzą hierarchię na jej szczycie jest proces `init`, który zawsze posiada identyfikator 1. Proces, który traci rodzica jest adoptowany przez `init`. Utrata rodzica najczęściej oznacza, że proces macierzysty zakończył pracę przed procesem potomnym.

Proces zombie to proces, który zakończył swoje działania, lecz informacja o im nadal jest przechowywana w systemie. Formalnie rzecz biorąc proces zombie powstaje gdy jego proces potomny zakończył działanie, a proces macierzysty nie odczytał jeszcze jego statusu zakończenia. Istnienie w systemie takich procesów powoduje blokowanie pewnych zasobów systemowych.

Procesy zombie wyróżniane są oznaczeniem `defunkt`

Polecenia wewnętrzne powłoki `bash` do działania na procesach: `jobs, fg, bg`.

W celu przedstawienia działanie polecenia `jobs` należy uruchomić kilka zadań. W tym przypadku uruchomiono edytor `vi` w tle poleceniem: `vi&`

```
[root@localhost ~]# vi&
[1] 870
[root@localhost ~]# jobs -l
[1]+   870 Zatrzymany (wyjście na tty)    vi
[root@localhost ~]#
```

Polecenie `jobs -l` podaje `PID` procesu i jego status `zatrzymany` (`stopped`).
Informacje na temat innych opcji polecenia: `hepl jobs`
Edytor vi można uczynić procesem pierwszoplanowym poprzez polecenie: `fg vi`.

Zadanie 8

W systemie uruchomić kilka zadań pierwszoplanowych i zadań w tle i udokumentować akcje statusu z zadań wykonywanych w tle na zadania pierwszoplanowe i na odwrót.

Wysyłanie sygnałów do procesów polecenie `kill`

Procesy komunikują się z jądrem systemu, a także między sobą, aby koordynować swoje działania. Linux wspiera kilka mechanizmów komunikacji zwanych IPC (Inter-Process Communication mechanisms). Jednym z nich są sygnały, zwane inaczej przerwaniami programowymi.Polecenia do tego służące mają następującą składnię:

```
kill [-sygnal] PID
```

Polecenie wysyła `-sygnal` do procesu o numerze `PID`. Wbudowane w powłokę `Bash` polecenie `kill` umożliwia także wysyłanie sygnałów do procesów identyfikowanych za pomocą numeru zadania `JID`.

Polecenie: `kill -l` wyświetla listę sygnałów, które `kill` może przesłać do procesów.

Najważniejsze sygnały:[10]

- jeżeli nie sprecyzujemy rodzaju sygnału `kill` wysyła sygnał `SIGINT` przerywający działanie procesu,
- `2 SIGINT` przerwanie procesu (ten sygnał jest wysyłany do procesu, gdy wciśniemy Ctrl+C),
- `9 SIGKILL` sygnał "zabicia" procesu,
- `19 SIGSTOP` zawieszenie procesu (Ctrl+Z) ,
- `18 SIGCONT` wznowienie zatrzymanego procesu

Np. polecenie:
`kill -9 3890`
wysyła sygnał `KILL` o numerze `9` do procesu o PID `3890`.

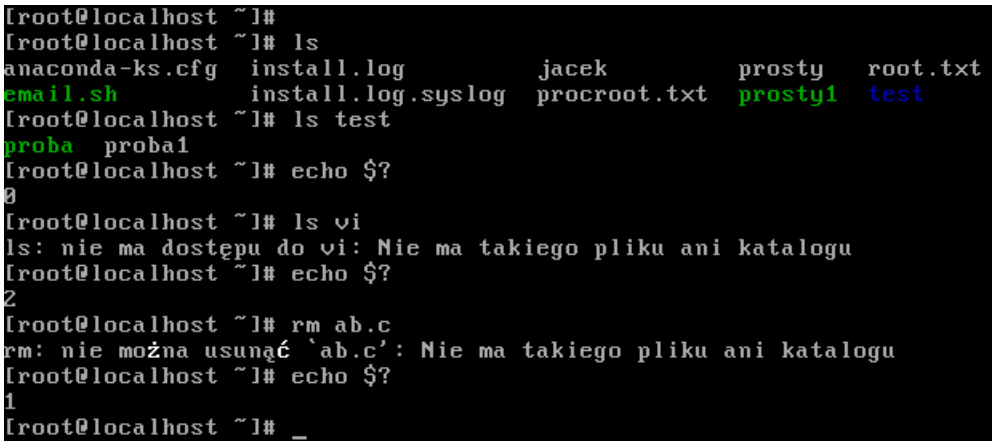
Zadanie 9

Udokumentuj kilka przykładów działania polecenia `kill` na wybrane procesy, korzystając z listy sygnałów dostępnej np. w referencji [10].

Status zakończenia procesu.

Każdy proces przekazuje do system informacje o tym jak zakończyło się przetwarzanie informacji jest to zmienna „`?`" zawierająca jedno bajtową liczbę tzw. status zakończenia procesu.

```
# echo $?
```

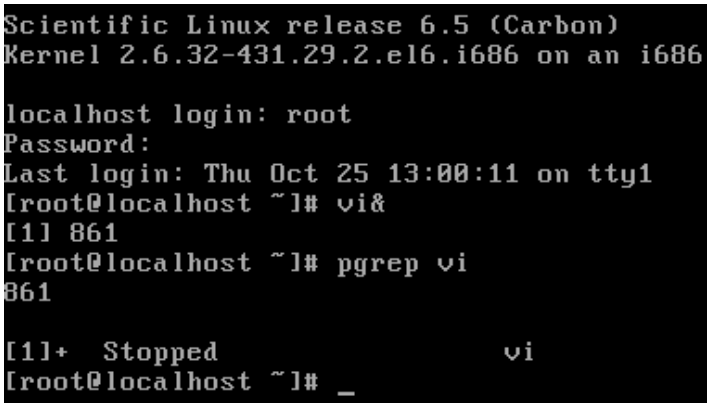


Wartość 0 oznacza poprawne zakończenie przetwarzania, wartość ≠ 0 oznacza błąd. Szczegółowy opis można znaleźć na stronie WWW [11]:

Wyszukiwanie procesów

Polecenie `pgrep` przegląda aktualnie uruchomione procesy i wyświetla identyfikatory procesów, które odpowiadają kryteriom wyboru.

Składnia: `pgrep [opcje] wyrażenie`



Najważniejsze opcje:
`-u` użytkownik zawęża wyniki poszukiwań do procesów danego użytkownika.
`-l` dodaje informację o nazwie procesu.

Literatura

[1] https://www.bezkompilatora.pl/jak-startuje-wspolczesny-linux/
[2] https://pl.wikibooks.org/wiki/Linux/Definicje/Runlevel

[3] https://hostovita.pl/blog/zarzadzanie-procesami-linux-ps-kill-nice/

[4] https://pl.wikibooks.org/wiki/Linux/Procesy

[5] Cezary Sobaniec, "System operacyjny. Linux – przewodnik użytkownika", wyd. NAKOM, 2002.

[6] PGID = "Process Group ID"; identyfikator grupy procesów, do której ten proces należy; na początku proces potomny należy do tej samej grupy co proces macierzysty (może on jednak założyć własną grupę procesów i stać się jej przywódcą, czyli może mieć PID=PGID). Można wysyłać sygnały do całych grup procesów a nie tylko do pojedynczych procesów:
https://mhanckow.students.wmi.amu.edu.pl/sop121/sop121b.htm
[7] https://github.com/freequaos/schedtool
[8] http://www.linux.pl/man/index.php?command=ps

http://www.techonthenet.com/linux/commands/ps.php

[9] https://www.nsa.gov/What-We-Do/Research/SELinux/

http://students.mimuw.edu.pl/SO/Projekt05-06/temat5-g7/se/se.html

[10] https://students.mimuw.edu.pl/SO/LabLinux/PROCESY/PODTEMAT_3/sygnaly.html
[11] http://www.freebsd.org/cgi/man.cgi?query=ps&manpath=SuSE+Linux/i386+11.3