Arkadiusz Sowa

USUX laboratorium 3

Procesy i sygnały

Zadanie 1

ps -Al : A- wybór wszystkich procesów, l- długi format, ps wyświetla listę uruchomionych procesów.

```
arek@arek-VirtualBox:~$ ps -Al
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S     0     1     0  0  80   0 - 25644 -      ?        00:00:02 systemd
1 S     0     2     0  0  80   0 -     0 -      ?        00:00:00 kthreadd
1 I     0     3     2  0  60 -20 -     0 -      ?        00:00:00 rcu_gp
1 I     0     4     2  0  60 -20 -     0 -      ?        00:00:00 rcu_par_gp
1 I     0     6     2  0  60 -20 -     0 -      ?        00:00:00 kworker/0:0H-kb
1 I     0     9     2  0  60 -20 -     0 -      ?        00:00:00 mm_percpu_wq
1 S     0    10     2  0  80   0 -     0 -      ?        00:00:00 ksoftirqd/0
1 I     0    11     2  0  80   0 -     0 -      ?        00:00:00 rcu_sched
1 S     0    12     2  0 -40   - -     0 -      ?        00:00:00 migration/0
```

F- flagi procesu, np. 4 oznacza że proces używał uprawnień superużytkownika;

S - stan procesu, np. S oznacza proces przerywalnie uśpiony;

UID- identyfikator użytkownika;

PID- identyfikator procesu;

PPID- identyfikator procesu-rodzica
C- procent użycia procesora przez proces;

PRI- priorytet procesu, czym większa wartość tego parametru tym mniejszy priorytet.

```
pri          PRI       priority of the process.  Higher number means
                       lower priority.
```

Jedną z najmniejszych wartości dla PRI (-40) posiadał u mnie proces ‘migration/0’

```
1 S     0    12     2  0 -40   - -     0 -      ?        00:00:00 migration/0
```

NI- pokazuje jak dużo czasu procesora jest przydzielane procesowi, wartości z przedziału -20 (najwyższy priorytet) do 19 (najniższy priorytet), operując wartością nice sterujemy priorytetem procesu.

```
ni           NI        nice value. This ranges from 19 (nicest) to -20
                       (not nice to others), see nice(1).  (alias nice).
```

ADDR- adres procesu;

SZ- zużycie pamięci;

WCHAN- nazwa funkcji jądra w której proces obecnie jest uśpiony;

TTY- kontrolujący terminal;

TIME- czas wykonywania procesu;

CMD- nazwa procesu.

Do przeglądania procesów konkretnego użytkownika możemy użyć **ps -lu user_name**.

Zadanie 2

Własne procesy pozwala wyświetlić przykładowo instrukcja ps z opcją -ux.

```
arek@arek-VirtualBox:~$ ps -ux
USER        PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
arek        796  0.0  0.2  21176 12040 ?        Ss   12:06   0:00 /lib/systemd/sy
arek        797  0.0  0.0 102868  3140 ?        S    12:06   0:00 (sd-pam)
arek        830  0.0  0.4 1411752 18628 ?       S<sl 12:06   0:00 /usr/bin/pulsea
arek        832  0.0  0.1 242584  6976 ?        SLl  12:06   0:00 /usr/bin/gnome-
```

Drzewo procesów od **PID=1** można wyświetlić za pomocą poniższej komedy:

```
arek@arek-VirtualBox:~$ pstree 1 -u
systemd─┬─ModemManager───2*[{ModemManager}]
        ├─NetworkManager───2*[{NetworkManager}]
        ├─accounts-daemon───2*[{accounts-daemon}]
        ├─acpid
        ├─avahi-daemon(avahi)───avahi-daemon
        ├─boltd───2*[{boltd}]
        ├─colord(colord)───2*[{colord}]
        ├─cron
        ├─cups-browsed───2*[{cups-browsed}]
```

Zadanie 3

Obsługę procesów sprawujemy za pomocą komendy kill, która wysyła sygnał do procesu o numerze PID podanym jako jeden z argumentów kill.

kill -s int numer_PID – zamyka proces o podanym numerze PID

kill -s stop numer_PID – zatrzymuje proces o podanym numerze PID

Należy pamiętać, iż nie możemy zamknąć procesu, który jest aktualnie zatrzymany, skutecznie udało mi się kilku krotnie zaciąć działanie całego systemu próbując zamknąć zatrzymany sposób na kilka sposobów. Aby wyłączyć taki proces należy go najpierw uruchomić (kill -s cont numer-PID).

Zmiana domyślnej obsługi sygnału SIGINT

```
arek@arek-VirtualBox:~$ trap "echo TEAZ_WIDZIMY_TO" SIGINT
arek@arek-VirtualBox:~$ ^CTEAZ_WIDZIMY_TO
```

Ignorowanie sygnału SIGINT (terminal nieczuły na ctrl +C)

```
arek@arek-VirtualBox:~$ trap '' SIGINT
arek@arek-VirtualBox:~$
```

Zadanie 4

jobs pokazuje aktualnie odbywające się procesy

```
arek@arek-VirtualBox:~$ jobs
[1]   Running                 sleep 200 &
[2]   Running                 sleep 180 &
[3]-  Running                 sleep 150 &
[4]+  Running                 sleep 100 &
```

fg pozwala przenieść ostatni z procesów na liście w jobs z tła do procesów działających w pierwszym planie. Brak & przy procesie sleep 100. Dodatkowo zatrzymałem proces poprzez użycie ctrl +z

```
arek@arek-VirtualBox:~$ fg
sleep 100
^Z
[4]+  Stopped                 sleep 100
arek@arek-VirtualBox:~$ jobs
[1]   Running                 sleep 200 &
[2]   Running                 sleep 180 &
[3]-  Running                 sleep 150 &
[4]+  Stopped                 sleep 100
```

bg działa odwrotnie do fg przenosząc proces z pierwszego planu do tła.

```
arek@arek-VirtualBox:~$ bg
[4]+ sleep 100 &
arek@arek-VirtualBox:~$ jobs
[1]   Running                 sleep 200 &
[2]   Running                 sleep 180 &
[3]-  Running                 sleep 150 &
[4]+  Running                 sleep 100 &
```

Ciekawymi opcjami funkcji jobs są -l (wyświetlane jest więcej informacji, między innymi PID) i -n (wyświetlanie tylko tych procesów, którego zmieniły status od ostatniego wywołania jobs).

Zadanie 5

Początkowo stworzyłem proces z parametrem nice 1

```
0 S  1000  9120  7593  0  81   1 -  2649 hrtime pts/0    00:00:00 sleep
```

Następnie zmieniłem nice na wartość 15

```
arek@arek-VirtualBox:~$ renice 15 9120
9120 (process ID) old priority 1, new priority 15
```

```
0 S  1000  9120  7593  0  95  15 -  2649 hrtime pts/0    00:00:00 sleep
```

Wraz ze wzrostem wartości NICE rośnie również wartość parametru PRI, a priorytet procesu spada.

Priorytetów nie można zmieniać dowolnie, wartość nice można określać w przedziale -19...20. Aby nadawać wartości ujemne parametrowi NICE trzeba posiadać uprawnienia superużytkownika.