Системне програмне забезпечення кіберфізичних систем

Лабораторна робота № 2

НАЗВА: Моніторинг та управління обчислювальними процесами в ОС Linux.
МЕТА: Навчитись роботі з командами та системними утилітами моніторингу та управління обчислювальними процесами в ОС Linux.

1. Загальні відомості

1.1. Віртуальна файлова система **procfs** (/proc file system) призначена для відображення інформації про обчислювальні процеси та систему в цілому. Вона розміщується в оперативній пам'яті та монтується у каталог /proc під час завантаження системи. Файлову систему **procfs** можна розглядати як інтерфейс доступу до внутрішніх структур даних ядра ОС Linux. Файлову систему **procfs** зручно використовувати при відлагодженні роботи обчислювальних процесів, вирішенні завдань системного адміністрування та дослідженні роботи системи з точки зору забезпечення її безпеки. Вміст **procfs** можна переглянути за допомогою команди: `$ ls -F /proc`

1.2. Кожний процес у /proc має свій підкаталог /proc/PID, де PID - це ідентифікатор процесу. В /proc/PID зібрана детальна інформація про відповідний обчислювальний процес. Переглянути список інформаційних параметрів процесу можна командою: `$ ls -F /proc/299` (299 - PID процесу). Переглянути значення окремого інформаційного параметра можна команою `$ cat /proc/<PID>/<parameter>`. Наприклад, команда `$ cat /proc/2962/cmdline` поверне командний рядок (повна назва виконавчого файла та аргументи командного рядку), яким був запущений на виконання процес з ідентифікатором 2962. Серед іншої в такий спосіб можна дізнатися наступну інформацію про процес:
/proc/PID/status - інформація про статус виконання процесу (в тому числі біжучий стан процесу та загальна інформація про його пам'ять);
/proc/PID/cmdline - командний рядок, яким був запущений процес;
/proc/PID/cwd - лінк на робочу директорію процесу (де міститься його виконавчий файл і з якої він був запущений на виконання);
/proc/PID/environ - назви та значення змінних оточення для даного процесу;
/proc/PID/exe - лінк на виконавчий файл процесу (якщо він ще існує на момент виконання процесу);
/proc/PID/limits - інформація про обмеження на використання системних ресурсів, що застосовуються до процесу;
/proc/PID/fd - директорія, яка містить лінки на всі дескриптори файлів відкриті процесом;
/proc/PID/task - директорія, яка містить лінки на всі задачі (tasks), які були запущені на виконання даним процесом.

Приклади:

1) визначити біжучий стан процесу:

```
$ cat /proc/299/status | grep State
```

2) визначити кількість програмних потоків (threads) процесу:
```
$ cat /proc/2091/status | grep Threads
```

3) визначити максимальну кількість файлів, які процес може відкрити одночасно:
```
$ cat /proc/299/limits | grep files
```

4) визначити кількість файлових дескрипторів, відкритих процесами веб-браузера firefox:
```
$ sudo ls -l /proc/$(pgrep firefox)/fd | wc -l
```
(команда pgrep повертає ідентифікатори процесів (pid), які відповідають вказаній назві виконавчого файлу чи іншим атрибутам процесу).

1.3. У файловій системі **procfs** також міститься важлива інформація про систему, зокрема:
/proc/cmdline – інформація про командний рядок ядра ОС;
/proc/cpuinfo - інформація про процесор;
/proc/console – інформація про відкриті на даний час консолі;
/proc/devices – список пристроїв з символьним та блочним вводом/виводом, які на даний час є в системі;
/proc/diskstats - інформація про логічні диски та розділи;
/proc/filesystems – файлові системи, які підтримуються системою;
/proc/interrupts - інформація про переривання;
/proc/loadavg – середнє завантаження системи (за останні 1, 5 та 15 хвилин);
/proc/locks – файли, які на даний момент залочені ядром ОС;
/proc/meminfo – інформація про використання основної пам’яті;
/proc/modules – завантажені модулі ядра;
/proc/partitions – список логічних розділів з параметрами;
/proc/stat – статистичні дані про роботу системи від часу її останнього запуску;
/proc/swaps – інформація про логічні розділи для своппінгу та їх параметри;
/proc/sys - інформація про параметри ядра (див. л/р №1);
/proc/uptime – час роботи системи з моменту старту і сумарний час простою усіх ядер процесора (в секундах);
/proc/version – інформація про систему (версія ядра та ін.).

Приклади:

1) визначити кількість ядер процесора:
```
$ cat /proc/cpuinfo | grep processor | wc -l
```

2) вивести список криптографічних модулів, які доступні в системі:
```
$ cat /proc/crypto | grep name
```

1.5. Основні команди управління обчислювальними процесами в ОС Linux наступні: **ps** - вивести список процесів та інформацію про них; **pstree** - вивести дерево процесів; **kill** - надіслати вказаний сигнал процесу із вказаним ідентифікатором (pid); **pgrep** -

повернути ідентифікатори процесів, які відповідають вказаній назві виконавчого файлу чи іншим атрибутам процесу; **pkill** - надіслати сигнал всім процесам, які відповідають вказаній назві виконавчого файлу чи іншим атрибутам процесу; **top** - інтерактивна команда, яка виводить загальну інформацію про стан системи та список найактивніших процесів. Команди **kill** та **pkill** по замовченню надсилають процесам сигнал SIGTERM, що призводить до завершення процесів.

1.6. **htop** - більш потужна і зручна утиліта, яка розширює функціональність команди top. **htop** відображає в режимі реального часу інформацію про систему та обчислювальні процеси, а також дозволяє виконувати операції по управлінню процесами (надіслати процесу сигнал, зупинити процес, зменшити чи збільшити пріоритет процесу). Утиліта **htop** відображає: 1) інформацію про біжуче завантаження ядер процесора, використання оперативної пам'яті (Mem) та розділу своппінгу (Swm); 2) кількість процесів (Tasks) та програмних потоків (thr), середню завантаженність системи за останні 1, 5 та 15 хвилин, час роботи системи з моменту старту (uptime); 3) список процесів, який можна відсортувати за обраним критерієм (команда F6) або представити у вигляді дерева процесів (команда F5). По замовченню процеси у списку відсортовані за зменшенням проценту використання процесорного часу.

1.7. Для моніторингу продуктивності роботи системи з точки зору виконання обчислень можна скористатись двома утилітами з набору утиліт sysstat (system statistics): 1) **mpstat** - утиліта для визначення статистичних даних про завантаження процесора (процесорів, ядер процесора) обчисленнями; 2) **pidstat** - утиліта для визначення статистичних даних про виконання окремих процесів.

Приклади:
1) отримати статистичні дані про використання всіх ядер процесора:

```
$ mpstat -P ALL
Linux 4.9.0-11-amd64 (b833)   02/29/2019     _x86_64_       (4 CPU)

10:18:44 PM  CPU    %usr   %nice    %sys %iowait    %irq   %soft  %steal  %guest  %gnice   %idle
10:18:44 PM  all    2.64    0.00    0.65    0.51    0.00    0.01    0.00    0.00    0.00   96.19
10:18:44 PM    0    2.56    0.01    0.73    0.42    0.00    0.02    0.00    0.00    0.00   96.26
10:18:44 PM    1    2.60    0.00    0.54    0.37    0.00    0.01    0.00    0.00    0.00   96.48
10:18:44 PM    2    2.69    0.00    0.66    0.55    0.00    0.02    0.00    0.00    0.00   96.07
10:18:44 PM    3    2.70    0.00    0.65    0.70    0.00    0.00    0.00    0.00    0.00   95.95
```

(%usr - процент часу виконання обчислень в режимі користувача, %nice - процент часу виконання обчислень в режимі користувача із зменшеним пріоритетом, %sys - процент часу виконання обчислень в режимі ядра, %iowait - процент простою процесора/ядра в очікуванні операцій вводу/виводу, %irq - процент часу витраченого процесором/ядром на обробку апаратних переривань, %soft - процент часу витраченого процесором/ядром на обробку програмних переривань, %idle - процент часу, на протязі якого процесори/ядра не виконували обчислень)

2) отримати статистичні дані про виконання процесу з ідентифікатором 2091:

```
$ pidstat -p 2091
Linux 4.9.0-11-amd64 (b833)       02/29/2019     _x86_64_      (4 CPU)
```

```
10:30:29 PM   UID       PID    %usr %system  %guest    %CPU   CPU  Command
10:30:29 PM  1000      2091    0.00    0.00    0.00    0.00     1  oosplash
```

3) отримати статистичні дані про виконання процесу з назвою виконавчого файлу firefox:

$ pidstat -G firefox

```
Linux 4.9.0-11-amd64 (b833)   02/29/2019     _x86_64_       (4 CPU)

10:39:16 PM   UID       PID    %usr %system  %guest    %CPU   CPU  Command
10:39:16 PM  1000      2312    1.70    0.33    0.00    2.03     0  firefox-bin
```

4) визначити процеси, які диспетчеризуються в режимі реального часу, їх пріоритети (prio) та режим диспетчеризації (policy):

$ pidstat -R

```
Linux 4.9.0-11-amd64 (b833)   02/29/2019     _x86_64_       (4 CPU)

10:36:55 PM   UID       PID prio policy  Command
10:36:55 PM     0         9   99   FIFO  migration/0
10:36:55 PM     0        11   99   FIFO  watchdog/0
10:36:55 PM     0        14   99   FIFO  watchdog/1
10:36:55 PM     0        15   99   FIFO  migration/1
10:36:55 PM     0        20   99   FIFO  watchdog/2
10:36:55 PM     0        21   99   FIFO  migration/2
10:36:55 PM     0        26   99   FIFO  watchdog/3
10:36:55 PM     0        27   99   FIFO  migration/3
10:36:55 PM     0       295   50   FIFO  irq/28-mei_me
10:36:55 PM     0       296    1   FIFO  i915/signal:0
10:36:55 PM     0       297    1   FIFO  i915/signal:1
10:36:55 PM     0       298    1   FIFO  i915/signal:2
10:36:55 PM     0       299    1   FIFO  i915/signal:4
10:36:55 PM     0       345   50   FIFO  irq/31-iwlwifi
10:36:55 PM  1000      1729    0   IDLE  tracker-miner-u
10:36:55 PM  1000      1735    0   IDLE  tracker-miner-a
```

1.8. Для обмеження використання системних ресурсів обчислювальними процесами можна скористатись системними утилітами ulimit та cpulimit. Утиліта **ulimit** призначена для визначення та встановлення обмежень на використання системних ресурсів обчислювальними процесами окремого користувача. Утиліта **cpulimit** призначена для встановлення обмежень на використання процесора вказаним процесом.

Приклади:

1) Визначити обмеження на використання системних ресурсів для даного користувача:

$ ulimit -a

```
core file size          (blocks, -c) 0
data seg size           (kbytes, -d) unlimited
scheduling priority             (-e) 0
file size               (blocks, -f) unlimited
pending signals                 (-i) 14572
max locked memory       (kbytes, -l) 64
max memory size         (kbytes, -m) unlimited
open files                      (-n) 1024
pipe size            (512 bytes, -p) 8
POSIX message queues     (bytes, -q) 819200
real-time priority              (-r) 0
stack size              (kbytes, -s) 8192
cpu time               (seconds, -t) unlimited
max user processes              (-u) 14572
virtual memory          (kbytes, -v) unlimited
file locks                      (-x) unlimited
```

2) Змінити максимальну кількість процесів, які користувач може запустити одночасно:

```
$ ulimit -u 14000
```

(для виконання команди можуть знадобитись права адміністратора)

3) обмежити використання процесорного часу процесом з ідентифікатором 2091 до 70%:
```
$ sudo cpulimit --pid 2091 --limit 70
```

2. Послідовність виконання роботи

2.1. Ознайомитись з відомостями про моніторинг та управління обчислювальними процесами в ОС Linux.
2.2. Дослідити роботу віртуальної файлової системи procfs. Навчитись визначати інформаційні параметри процесів та отримувати інформацію про систему за домомогою procfs.
2.3. Отримати інформацію про обмеження на використання системних ресурсів, що застосовуються до процесу bash.
2.4. Визначити кількість файлових дескрипторів, відкритих процесами bash.
2.5. Визначити кількість ядер процесора та вивести список криптографічних модулів, які доступні в системі.
2.6. Дослідити роботу команд управління обчислювальними процесами. Дослідити роботу утиліти htop.
2.7. Дослідити роботу системних утиліт mpstat та pidstat. Отримати статистичні дані про використання всіх ядер процесора. Отримати статистичні дані про виконання процесу з назвою виконавчого файлу bash.
2.8. Дослідити роботу системних утиліт ulimit та cpulimit. Визначити обмеження на використання системних ресурсів для даного користувача. Обмежити використання процесорного часу процесом bash до 50%.
2.9. Скласти та захистити звіт з лабораторної роботи.

3. Зміст звіту

3.1. Результати виконання завдань по визначенню інформаційних параметрів процесів та отриманню інформації про систему за допомогою procfs.
3.2. Результати дослідження роботи утиліти htop.
3.3. Результати виконання завдань по дослідженню роботи утиліт mpstat та pidstat.
3.4. Результати виконання завдань по дослідженню роботи утиліт ulimit та cpulimit.

4. Контрольні питання

4.1. Віртуальна файлова система procfs (/proc file system).
4.2. Визначення інформаційних параметрів процесу за домогою procfs.
4.3. Отримання інформації про систему за домомогою procfs.
4.4. Основні команди управління обчислювальними процесами в ОС Linux.

4.5. Призначення та використання системної утиліти htop.
4.6. Призначення та використання системної утиліти mpstat.
4.7. Призначення та використання системної утиліти pidstat.
4.8. Призначення та використання системної утиліти ulimit.
4.9. Призначення та використання системної утиліти cpulimit.

5. Джерела

1. Daniel P. Bovet, Marco Cesati, Understanding the Linux Kernel, 3rd edition, O'Reilly Media, 2005. - 944 p.

2. Robert Love, Linux Kernel Development, 3rd edition, Addison-Wesley Professional, 2010. - 440 p.

3. The Linux Kernel documentation, www.kernel.org/doc/html/latest/

4. The /proc file system (Linux Documentation), https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/Documentation/filesystems/proc.txt