Narzędzia zarządzania i monitorowania w systemach Linux

Dariusz Dwornikowski

Institute of Computing Science
Poznań University of Technology

Zarządzanie sieciami komputerowymi.

Plan zajęć

- System plików procfs i sysfs
- Narzędzia monitorowania systemu Linux
- Narzędzia zarządzania systemu Linux
- Narzędzia zarządzania procesami supervisord i monit

System plików procfs

Specjalny system plików w systemach rodziny UNIX do prezentacji informacji o procesach oraz systemie. Jest to plikowy interfejs do jądra do informacji o każdym procesie obecnym w systemie. Można go znaleźć w punkcie montowania /proc. Oprócz tychinformcji zawiera także inne dane, nie bezpośrednio związane z procesami (głównie /proc/sys).

System plików procfs

```
tdi@blackstar ~ $ ls /proc/
1       225     310     31986   368     678     7032    793         execdomains  pagetypeinfo
1005    233     311     31994   37      679     708     8           fb           partitions
106     237     313     31995   38      683     709     80          filesystems  sched_debug
107     24857   31560   32151   383     6865    712     81          fs           schedstat
11619   24863   31561   32153   3872    6867    715     840         interrupts   scsi
11687   24865   31567   32154   393     6868    727     844         iomem        self
12      25      31569   32280   41      687     731     96          ioports      slabinfo
120     26      31619   32284   43      6870    734     acpi        irq          softirqs
124     2601    31620   32287   44      6876    74      asound      kallsyms     stat
134     262     31621   32297   486     6877    741     buddyinfo   kcore        swaps
141     269     31622   32370   5       6905    742     bus         key-users    sys
16      27      31623   32576   516     6920    744     cgroups     kmsg         sysrq-trigger
16259   28      31624   32622   528     6926    747     cmdline     kpagecount   sysvipc
16261   286     31626   32685   543     6934    748     config.gz   kpageflags   timer_list
16268   287     31654   32717   545     6941    749     consoles    loadavg      timer_stats
17      29      31656   33      611     6947    75      cpuinfo     locks        tty
2       292     31689   339     626     6961    757     crypto      meminfo      uptime
212     293     317     34      649     6985    758     devices     misc         version
213     298     31753   35      660     7       76      diskstats   modules      vmallocinfo
217     3       318     36      661     700     77      dma         mounts       vmstat
22      30      319     360     669     7000    78      dri         mtrr         zoneinfo
221     308     31966   364     670     701     79      driver      net
tdi@blackstar ~ $
```

Struktura /proc/PID/status

Przykład monitorowania procesu aktualnego: `cat /proc/self/status`.

Ważniejsze pola

State R - running, S - sleeping, D - sleeping in wait, Z - zombie, T - traced lub stopped

Pid,PPid,Uid,Gid PID, parent PID, struktura UID, struktura GID

name nazwa wykonywalnego pliku (np. cat)

VmSize cały rozmiar programu w pamięci

VmData,VmPeak odpowiednio: rozmiar pamięci dla danych, stosu i textu oraz rozmiar maksymalny w pamięci

Threads liczba wątków

Cpus_allowed liczba dozwolonych procesorów

Cwiczenie

Spróbuj wyświetlić jednym poleceniem status dla procesu init.

Struktura /proc/PID/status

Przykład monitorowania procesu aktualnego: `cat /proc/self/status`.

Ważniejsze pola

State R - running, S - sleeping, D - sleeping in wait, Z - zombie, T - traced lub stopped

Pid,PPid,Uid,Gid PID, parent PID, struktura UID, struktura GID

name nazwa wykonywalnego pliku (np. cat)

VmSize cały rozmiar programu w pamięci

VmData,VmPeak odpowiednio: rozmiar pamięci dla danych, stosu i textu oraz rozmiar maksymalny w pamięci

Threads liczba wątków

Cpus_allowed liczba dozwolonych procesorów

Cwiczenie

Spróbuj wyświetlić jednym poleceniem status dla procesu init.

```
cat /proc/1/status
```

Inne informacje w /proc/PID/

stat to samo co status ale w formie parsowalnej

environ zmienne środowiskowe

cmdline argumenty linii poleceń

cpu aktualny i ostanio używany procesor

cwd katalog pracy (working directory)

exe link do pliku wykonywalnego

sched statystyki schedulera

io statystyki I/O

Struktury niepidowe w /proc

Oprócz informacji o procesach, procfs zawiera także informacje o systemie. W dawnych wersjach jądra było tych informacji więcej, jednak część (np. urządzenia) zostały przeniesione do sysfs (/sys) aby zapobiec zaśmieceniu /proc. Nadal pozostały jednak pewne ciekawe informacje, które można wyciągnąć oraz parametry, które można ustawić.

/proc/loadavg informacje o load

/proc/meminfo informacje o pamięci

/proc/vmstat informacje o pamięci wirtualnej

/proc/version wersja jądra

/proc/uptime uptime systemu

/proc/modules lista załadowanych modułów

Struktury niepidowe w /proc

Oprócz informacji o procesach, procfs zawiera także informacje o systemie. W dawnych wersjach jądra było tych informacji więcej, jednak część (np. urządzenia) zostały przeniesione do sysfs (/sys) aby zapobiec zaśmieceniu /proc. Nadal pozostały jednak pewne ciekawe informacje, które można wyciągnąć oraz parametry, które można ustawić.

/proc/loadavg informacje o load

/proc/meminfo informacje o pamięci

/proc/vmstat informacje o pamięci wirtualnej

/proc/version wersja jądra

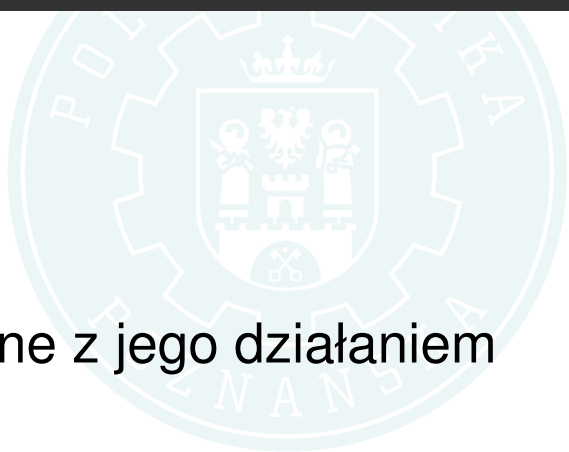
/proc/uptime uptime systemu

/proc/modules lista załadowanych modułów

Cwiczenie

Co jeszcze można tam wyczytać ?

Podkatalog /proc/sys



Tutaj można odczytać i zmienić paremetry jądra związane z jego działaniem wewnętrznym oraz związanym np. z obsługą sieci.

Ważne katalogi

/proc/sys/kernel globalne informacje jądra, tuning, itd

/proc/sys/net stosy sieciowe

/proc/sys/fs systemy plików (np. quota)

Katalog /proc/sys/kernel

panic czas w skundach, po których kernel restartuje się (reboot) po panic (0 domyślnie)

panic_on_oops czy oops wywoła panic (0 lub 1)

osrelease wydanie dystrybucji

hostname host name (lol)

random/poolsize wielkość puli entropii

random/entropy_avail dostępna pula

Katalog /proc/sys/net

ipv4 statystyki stosu ipv4

ipv6 statystyki stosu ipv6

netfilter ustawienia i statystyki firewalla

/proc/sys/net/ipv4/ip_forward 1 - ruter, 0 - desktop

/proc/sys/net/ipv4/ip_default_ttl domyślny TTL

/proc/sys/net/ipv4/tcp_keepalive_time keepalive dla TCP

/proc/sys/net/conf/IFACE/rp_filter reverse path filtering, trasowanie pakietów innych sieci

/proc/sys/net/route/ ustawienia rutowania

Narzędzia korzystające z /proc

top, htop, atop, glances narzędzia do pokazywania statystyk procesów w real time

ps,pstree pakiet psutils pokazuje nam snapshot procesów, filtracja w man ps

iotop monitorowanie operacji I/O (pakiet sysstat)

iostat statystyki I/O

netstat statystki sieci

ss statystyki gniazd sieciowych (pakiet iproute)

vmstat monitorowanie pamięci

uptime uptime ;)

w kto zalogowany + load

free ile wolnej pamięci

pmap zajętośc pamięci dla PID (pmap -d PID)

iptraf kolorowe statystyki sieci (trzeba instalować)

lsof otwarte deskryptory i kto je ma

Przykłady użycia

`ps aux` wszystkie procesy
`ps -AlF` extra full mode
`netstat -anpt` połączenia, gniazda TCP
`pmap -d 1` pamięc zużywana przez PID 1
`vmstat -f` liczba forków od uruchomienia

System plików sysfs

System plików w pamięci (ramfs) służący do prezentacji struktur danych jądra, urządzeń itd. Punkt montowania to sys.

devices urządzenia

  bus szyny dostępne w komputerze

block urządzenia blokowe odkryte w systemie

class klasy urządzeń np. grafika, drukarka, dźwięk

platform widok dla danej platformy sprzętowej

Supervisord i monit

Programy supervisord i monit służą do zarządzania procesami, ich uruchamiania, pilnowania ich, restartowania. Przydatne są do uruchamiania programów userspace (np. serwery WSGI, FastCGI, skrypty a'la cron). Różnice:

monit monit uruchamia program zewnętrzny, np. wywołując skrypt z init.d, pilnuje czy istnieje plik pid. Monit dodatkowo ma możliwość sprawadzania pewnych warunków (np. load, zajętość pamięci)

supervisord uruchamia program jako swoje dziecko, ma całkowitą kontrolę nad procesami

Konfiguracja supervisord

W pliku supervisord.conf wpisujemy:

```
[program:top]
command = /bin/top ; sciezka do "binarki"
numprocs = 2 ; ile ma byc procesow
directory = /tmp ; CWD
user = tdi ; jako jaki user
exitcodes = 0,2
autorestart = true ; czy restartowac
stdout_logfile=/tmp/out
```

Supervisord cli

Supervisord udostępnia zarządzanie przez CLI:

```
supervisorctl> status
supervisorctl> avail
supervisorctl> start/stop/restart
supervisorctl> add ...
```

Oraz webowo:

```
[inet_http_server]
port=127.0.0.1:9001
username=user
password=123
```

Supervisord cli

Supervisord udostępnia zarządzanie przez CLI:

```
supervisorctl> status
supervisorctl> avail
supervisorctl> start/stop/restart
supervisorctl> add ...
```

Oraz webowo:

```
[inet_http_server]
port=127.0.0.1:9001
username=user
password=123
```

Cwiczenia

Zainstaluj i skonfiguruj supervisord dla dowolnego procesu.

Konfiguracja monit

Konfiguracja w /etc/monitrc

```
check process httpd with pidfile /tmp/http.pid
   start program = "/etc/init.d/httpd start"
   stop program = "/etc/init.d/httpd stop"
   if failed port 80 then restart
   if 3 restarts within 4 cycles then timeout
```

Testowac mozemy na http://localhost:2812/

Dodatkowe reguły

Można stosować dodatkowe reguły. Składnia reguły: `http://mmonit.com/monit/documentation/monit.html#service_tests`

```
if 3 restarts within 4 cycles then timeout
if cpu > 80% for 5 cycles then restart
if children > 200 then restart
if loadavg(5min) greater than 10 for 8 cycles then stop
if uptime > 3 days then exec "myscript.sh"
if failed host 1.2.3.4 port 3333 protocol ssh then alert
```