9 Notes on selecting processes in proc.mem and proc.num items

Processes modifying their commandline

Some programs use modifying their commandline as a method for displaying their current activity. A user can see the activity by running `ps` and `top` commands. Examples of such programs include *PostgreSQL*, *Sendmail*, *Zabbix*.

Let's see an example from Linux. Let's assume we want to monitor a number of Zabbix agent processes.

`ps` command shows processes of interest as

```
$ ps -fu zabbix
UID        PID  PPID  C STIME TTY          TIME CMD
...
zabbix    6318     1  0 12:01 ?        00:00:00 sbin/zabbix_agentd -c
/home/zabbix/ZBXNEXT-1078/zabbix_agentd.conf
zabbix    6319  6318  0 12:01 ?        00:00:01 sbin/zabbix_agentd:
collector [idle 1 sec]
zabbix    6320  6318  0 12:01 ?        00:00:00 sbin/zabbix_agentd: listener
#1 [waiting for connection]
zabbix    6321  6318  0 12:01 ?        00:00:00 sbin/zabbix_agentd: listener
#2 [waiting for connection]
zabbix    6322  6318  0 12:01 ?        00:00:00 sbin/zabbix_agentd: listener
#3 [waiting for connection]
zabbix    6323  6318  0 12:01 ?        00:00:00 sbin/zabbix_agentd: active
checks #1 [idle 1 sec]
...
```

Selecting processes by name and user does the job:

```
$ zabbix_get -s localhost -k 'proc.num[zabbix_agentd,zabbix]'
6
```

Now let's rename `zabbix_agentd` executable to `zabbix_agentd_30` and restart it.

`ps` now shows

```
$ ps -fu zabbix
UID        PID  PPID  C STIME TTY          TIME CMD
...
zabbix    6715     1  0 12:53 ?        00:00:00 sbin/zabbix_agentd_30 -c
/home/zabbix/ZBXNEXT-1078/zabbix_agentd.conf
zabbix    6716  6715  0 12:53 ?        00:00:00 sbin/zabbix_agentd_30:
collector [idle 1 sec]
zabbix    6717  6715  0 12:53 ?        00:00:00 sbin/zabbix_agentd_30:
listener #1 [waiting for connection]
zabbix    6718  6715  0 12:53 ?        00:00:00 sbin/zabbix_agentd_30:
listener #2 [waiting for connection]
```

```
zabbix    6719  6715  0 12:53 ?        00:00:00 sbin/zabbix_agentd_30:
listener #3 [waiting for connection]
zabbix    6720  6715  0 12:53 ?        00:00:00 sbin/zabbix_agentd_30:
active checks #1 [idle 1 sec]
...
```

Now selecting processes by name and user produces an incorrect result:

```
$ zabbix_get -s localhost -k 'proc.num[zabbix_agentd_30,zabbix]'
1
```

Why a simple renaming of executable to a longer name lead to quite different result ?

Zabbix agent starts with checking the process name. `/proc/<pid>/status` file is opened and the line `Name` is checked. In our case the `Name` lines are:

```
$ grep Name /proc/{6715,6716,6717,6718,6719,6720}/status
/proc/6715/status:Name: zabbix_agentd_3
/proc/6716/status:Name: zabbix_agentd_3
/proc/6717/status:Name: zabbix_agentd_3
/proc/6718/status:Name: zabbix_agentd_3
/proc/6719/status:Name: zabbix_agentd_3
/proc/6720/status:Name: zabbix_agentd_3
```

The process name in `status` file is truncated to 15 characters.

A similar result can be seen with `ps` command:

```
$ ps -u zabbix
  PID TTY          TIME CMD
...
 6715 ?        00:00:00 zabbix_agentd_3
 6716 ?        00:00:01 zabbix_agentd_3
 6717 ?        00:00:00 zabbix_agentd_3
 6718 ?        00:00:00 zabbix_agentd_3
 6719 ?        00:00:00 zabbix_agentd_3
 6720 ?        00:00:00 zabbix_agentd_3
 ...
```

Obviously, that is not equal to our `proc.num[]` `name` parameter value `zabbix_agentd_30`. Having failed to match the process name from `status` file the Zabbix agent turns to `/proc/<pid>/cmdline` file.

How the agent sees the “cmdline” file can be illustrated with running a command

```
$ for i in 6715 6716 6717 6718 6719 6720; do cat /proc/$i/cmdline | awk
'{gsub(/\x0/,"<NUL>"); print};'; done
sbin/zabbix_agentd_30<NUL>-
c<NUL>/home/zabbix/ZBXNEXT-1078/zabbix_agentd.conf<NUL>
```

```
sbin/zabbix_agentd_30: collector [idle 1
sec]<NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL>...
sbin/zabbix_agentd_30: listener #1 [waiting for
connection]<NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL>...
sbin/zabbix_agentd_30: listener #2 [waiting for
connection]<NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL>...
sbin/zabbix_agentd_30: listener #3 [waiting for
connection]<NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL>...
sbin/zabbix_agentd_30: active checks #1 [idle 1
sec]<NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL>...
```

`/proc/<pid>/cmdline` files in our case contain invisible, non-printable null bytes, used to terminate strings in *C* language. The null bytes are shown as "<NUL>" in this example.

Zabbix agent checks "cmdline" for the main process and takes a `zabbix_agentd_30`, which matches our `name` parameter value `zabbix_agentd_30`. So, the main process is counted by item `proc.num[zabbix_agentd_30,zabbix]`.

When checking the next process, the agent takes `zabbix_agentd_30: collector [idle 1 sec]` from the `cmdline` file and it does not meet our `name` parameter `zabbix_agentd_30`. So, only the main process which does not modify its commandline, gets counted. Other agent processes modify their command line and are ignored.

This example shows that the `name` parameter cannot be used in `proc.mem[]` and `proc.num[]` for selecting processes in this case.

Using `cmdline` parameter with a proper regular expression produces a correct result:

```
$ zabbix_get -s localhost -k 'proc.num[,zabbix,,zabbix_agentd_30[ :]]'
6
```

Be careful when using `proc.mem[]` and `proc.num[]` items for monitoring programs which modify their commandlines.

Before putting `name` and `cmdline` parameters into `proc.mem[]` and `proc.num[]` items, you may want to test the parameters using `proc.num[]` item and `ps` command.

Linux kernel threads

Threads cannot be selected with ''cmdline'' parameter in ''proc.mem[]'' and ''proc.num[]'' items

Let's take as an example one of kernel threads:

```
$ ps -ef| grep kthreadd
root         2     0  0 09:33 ?        00:00:00 [kthreadd]
```

It can be selected with process `name` parameter:

```
$ zabbix_get -s localhost -k 'proc.num[kthreadd,root]'
```

```
1
```

But selection by process `cmdline` parameter does not work:

```
$ zabbix_get -s localhost -k 'proc.num[,root,,kthreadd]'
0
```

The reason is that Zabbix agent takes the regular expression specified in `cmdline` parameter and applies it to contents of process `/proc/<pid>/cmdline`. For kernel threads their `/proc/<pid>/cmdline` files are empty. So, `cmdline` parameter never matches.

Counting of threads in ''proc.mem[]'' and ''proc.num[]'' items

Linux kernel threads are counted by `proc.num[]` item but do not report memory in `proc.mem[]` item. For example:

```
$ ps -ef | grep kthreadd
root         2     0  0 09:51 ?        00:00:00 [kthreadd]
```

```
$ zabbix_get -s localhost -k 'proc.num[kthreadd]'
1
```

```
$ zabbix_get -s localhost -k 'proc.mem[kthreadd]'
ZBX_NOTSUPPORTED: Cannot get amount of "VmSize" memory.
```

But what happens if there is a user process with the same name as a kernel thread ? Then it could look like this:

```
$ ps -ef | grep kthreadd
root         2     0  0 09:51 ?        00:00:00 [kthreadd]
zabbix    9611  6133  0 17:58 pts/1    00:00:00 ./kthreadd
```

```
$ zabbix_get -s localhost -k 'proc.num[kthreadd]'
2
```

```
$ zabbix_get -s localhost -k 'proc.mem[kthreadd]'
4157440
```

`proc.num[]` counted both the kernel thread and the user process. `proc.mem[]` reports memory for the user process only and counts the kernel thread memory as if it was 0. This is different from the case above when ZBX_NOTSUPPORTED was reported.

Be careful when using `proc.mem[]` and `proc.num[]` items if the program name happens to match one of the thread.

Before putting parameters into `proc.mem[]` and `proc.num[]` items, you may want to test the parameters using `proc.num[]` item and `ps` command.