クラウド基盤構築演習

第一部
クラウド基盤を支えるインフラ技術
～ 第6回 Linux問題判別と内部構造演習

ver1.1 2012/05/01

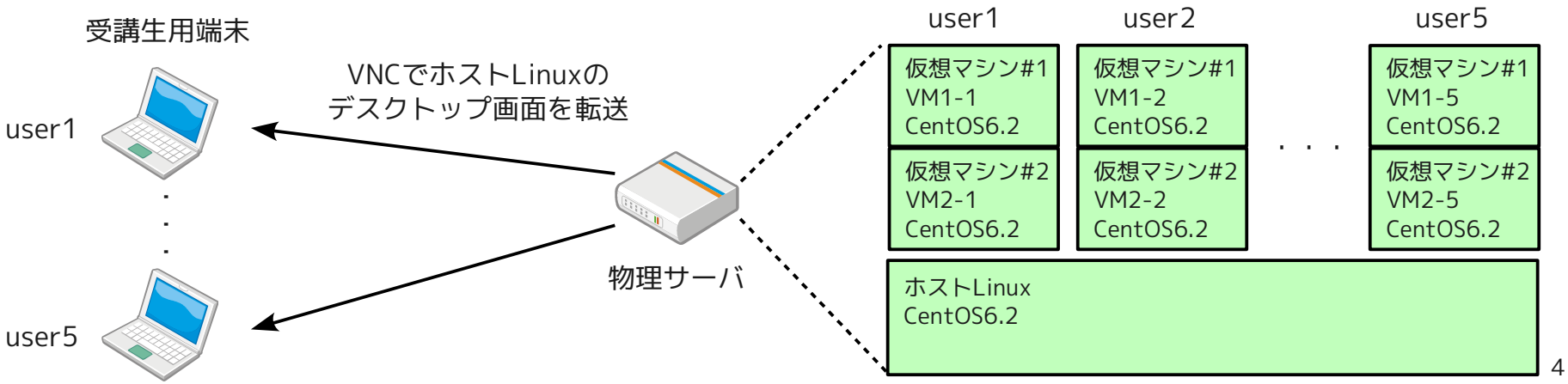
目次

- 演習環境の説明
- 問題判別基本コマンド演習
- Upstartジョブ管理演習
- プロセス管理/メモリ管理演習

演習環境の説明

演習環境 (1)

- 受講生（最大）5名ごとに演習用の物理サーバが割り当てられています。
  - 各受講生は、自分に割り当てられた「物理サーバ（IPアドレス）」と「ログインユーザ（user1～user5）」を確認してください。
- 各物理サーバには、ホストLinuxとして、CentOS6.2が導入されています。このホストLinuxのデスクトップ画面をVNCで受講生用端末に表示して演習を行います。
  - VNC接続の方法は、別途インストラクタよりガイドがあります。
- この演習では、Linux KVMによる仮想化環境を利用して、CentOS6.2をゲストOSとする仮想マシンを「受講生1名につき2台」作成します。
  - 各受講生は自分が作成する仮想マシンについて、「仮想マシン名、ホストネーム、IPアドレス」の割り当てルール（次ページ参照）を確認してください。



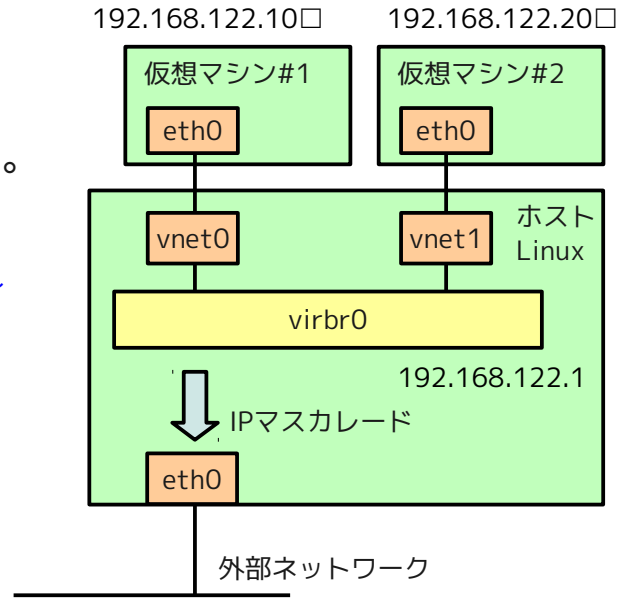
演習環境 (2)

- 演習で作成する仮想マシンは、ホストLinux上の仮想ブリッジによるプライベートネットワークに接続されます。
  - 仮想マシンから外部ネットワークには、IPマスカレードで接続します。外部ネットワークから仮想マシンに接続することはできません。
  - ホストLinuxから仮想マシンにログインすることは可能です。
- 仮想マシンを使用する際は、次のどちらかで接続します。
  - ホストLinuxで「virt-manager」を起動して、仮想マシンのコンソール画面を開く。
  - ホストLinuxから仮想マシンにSSHでログインする
  - 仮想マシン名、ホストネーム、IPアドレスは下表を使用します。
    □には、割り当てられたユーザ番号（1～5）が入ります。

※演習手順において、□で示された部分も同様にユーザ番号（1～5）を入れてください。

仮想マシン名	ホストネーム	IPアドレス /ネットマスク	デフォルトゲートウェイ
仮想マシン#1 VM1-□	vm1-□	192.168.122.10□ /255.255.255.0	192.168.122.1
仮想マシン#2 VM2-□	vm2-□	192.168.122.20□ /255.255.255.0	192.168.122.1

仮想ネットワークの構成



演習環境 (3)

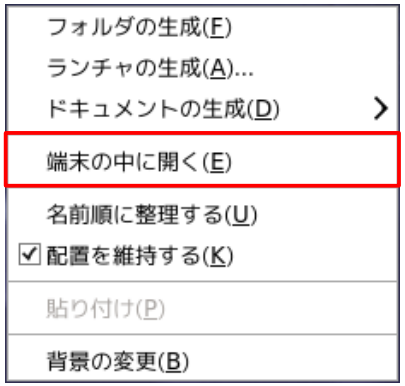
- ホストLinuxでコマンド端末を開くには、デスクトップを右クリックして「端末の中を開く」を選択します。

- 「virt-manager」を起動するには、コマンド端末で「virt-manager」を実行するか、デスクトップ左上の「アプリケーション」メニューから「システムツール→仮想マシンマネージャー」を選択します。

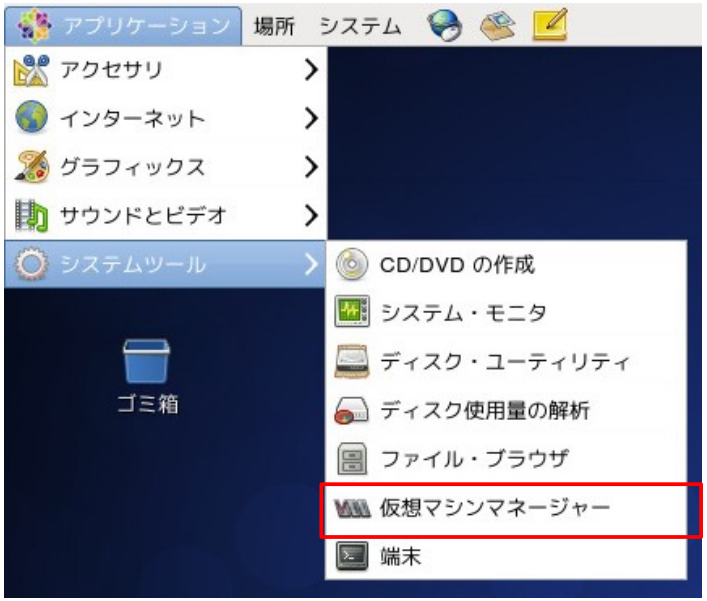
- 「Firefox」を起動するには、コマンド端末で「firefox」を実行するか、デスクトップ上部のアイコン（「システム」メニューの右横）をクリックします。

- ホストLinux上では、CentOS6.2のインストールメディアの内容がHTTPで公開されています。ホストLinuxのFirefoxから次のURLにアクセスして、内容を確認してください。
  - http://192.168.122.1/repo

デスクトップの
右クリックメニュー



デスクトップのアプリケーションメニュー



問題判別基本コマンド演習

演習内容

- この演習では、次の作業を行います。
  - 仮想マシン#1（VM1-□）において、sosreportコマンドで構成情報を取得します。
    - 収集した構成情報の内容を確認します。
  - 仮想マシン#1（VM1-□）において、システムログの収集に関する設定を行います。
    - 特定のFacilityのログを記録するログファイルを用意します。
    - カーネルメッセージを記録するログファイルを用意します。
  - 仮想マシン#1（VM1-□）において、問題判別に有用な情報をコマンドで収集します。

sosreportによる構成情報の収集 (1)

- 仮想マシン#1（VM1-□）において、sosreportコマンドで構成情報を取得します。
  - ホストLinuxのコマンド端末からVM1-□にログインします。

```
# ssh root@192.168.122.10□    ← ホストLinuxのコマンド端末で実行
```

  - sosreportを実行します。

```
[root@vm1-□ ~]# sosreport --report

sosreport (version 2.2)
. . .

ENTER を押して継続するか、又は CTRL-C で終了します。      ← [Enter]を入力

名前のイニシャルと姓を記入してください [vm1-□]:           ← [Enter]を入力
作成しているレポートのケース番号を記入してください :       ← [Enter]を入力

 プラグインを実行中です。少しお待ち下さい ...

  Completed [50/50] ...
アーカイブを圧縮しています...

sos レポートが生成されて、以下の場所に保存されました:
  /tmp/sosreport-vm1-1-20120517094556-aced.tar.xz

The md5sum is: 0e87cb212cf8f17480e1826606a7aced

このファイルを担当のサポート代表者に送信してください。
```

- 「--report」オプションを指定するとWebブラウザで閲覧するためのHTMLファイルが作成されます。

sosreportによる構成情報の収集 (2)

- VM1-□にhttpdサービスを未導入の場合は、ここで導入／起動しておきます。

```
[root@vm1-□ ~]# yum install httpd
・・・
================================================================================
 Package             Arch         Version                       Repository  Size
================================================================================
Installing:
 httpd               x86_64       2.2.15-15.el6.centos          base       809 k
Installing for dependencies:
 apr                 x86_64       1.3.9-3.el6_1.2               base       123 k
 apr-util            x86_64       1.3.9-3.el6_0.1               base        87 k
 apr-util-ldap       x86_64       1.3.9-3.el6_0.1               base        15 k
 httpd-tools         x86_64       2.2.15-15.el6.centos          base        70 k

Transaction Summary
================================================================================
Install       5 Package(s)

Total download size: 1.1 M
Installed size: 3.5 M
Is this ok [y/N]: y
・・・
Complete!
[root@vm1-□ ~]# chkconfig httpd on
[root@vm1-□ ~]# service httpd start
httpd を起動中: httpd: apr_sockaddr_info_get() failed for vm1-□
httpd: Could not reliably determine the server's fully qualified domain name, using
127.0.0.1 for ServerName
                                                           [  OK  ]
```

sosreportによる構成情報の収集 (2)

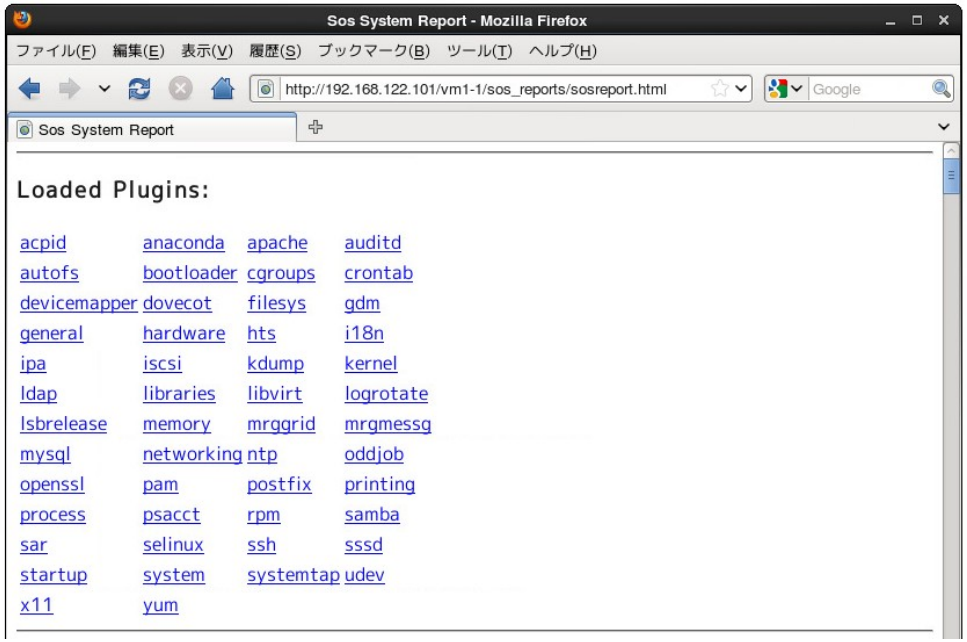
- 収集したファイルをWebブラウザで閲覧できるようにファイルを展開します。

```
[root@vm1-□ ~]# cd /var/www/html
[root@vm1-□ html]# tar -xvJf /tmp/sosreport-vm1-1-20120517094556-aced.tar.xz
vm1-1-201205170945133721536/
vm1-1-201205170945133721536/route
vm1-1-201205170945133721536/installed-rpms
vm1-1-201205170945133721536/ps
vm1-1-201205170945133721536/hostname
vm1-1-201205170945133721536/uptime
vm1-1-201205170945133721536/chkconfig
vm1-1-201205170945133721536/lspci
. . .
[root@vm1-□ html]# chown -R apache.apache vm1-□-201205170945133721536
[root@vm1-□ html]# ln -s vm1-1-201205170945133721536 vm1-□
```

- ファイル名後半の記号（タイムスタンプ/チェックサム）は環境によって異なります。

sosreportによる構成情報の収集 (3)

- ホストLinuxでFirefoxを起動して、次のURLを開きます。
  「http://192.168.122.10□/vm1-□/sos_reports/sosreport.html」



  - どのようなファイルがあるかブラウザから確認してください。シンボリックリンクファイルはブラウザから表示できない場合があります。そのような場合は、元のファイルを直接に開いて確認してください。

- 以上で「sosreportによる構成情報の収集」は完了です。

システムログの収集 (1)

- 仮想マシン#1（VM1-□）において、システムログの収集に関する設定を行います。
- 質問 1 : 次の設定方法を考えてください。
  - Facilityがlocal0のログメッセージが「/var/log/local0.log」に記録されるよう設定します。
  - loggerコマンドを利用して設定を確認します。
  - さらに、次のコマンドでログファイルをバックアップします。

```
# mv /var/log/local0.log /var/log/local0.log.bak
```

  - 再度 logger コマンドを使用して、メッセージが 「/tmp/local0.log.bak」に出力されることを確認します。
  - 新たに「/var/log/local0.log」にログが出力されるためにはどのような操作が必要でしょうか。

システムログの収集 (2)

- 質問１の解答例
  - ホストLinuxのコマンド端末からVM1-□にログインします。

```
# ssh root@192.168.122.10□    ← ホストLinuxのコマンド端末で実行
```

  - 設定ファイル「/etc/rsyslog.conf」に次の設定を追加します。

/etc/rsyslog.conf

```
# Save boot messages also to boot.log
local7.*                                                /var/log/boot.log

local0.*                                                /var/log/local0.log    ← この行を追加
```

  - rsyslogサービスをreloadして設定変更を反映します。

```
[root@vm1-□ ~]# service rsyslog reload
Reloading system logger...                                 [  OK  ]
```

  - loggerコマンドでFacilityがlocal0（Priorityはinfo）のログメッセージを出力して、「/var/log/local0.log」に記録されることを確認します。

```
[root@vm1-□ ~]# logger -p local0.info -t 'Test' 'Hello, World!'
[root@vm1-□ ~]# cat /var/log/local0.log
Jan 19 19:48:39 vm1-□ Test: Hello, World!
```

システムログの収集 (3)

- pgrepコマンドで「rsyslogd」のプロセスIDを確認して、lsofコマンドでこのプロセスがオープンしているファイルを確認します。

```
[root@vm1-□ ~]# pgrep rsyslogd
1159
[root@vm1-□ ~]# lsof -p 1159
COMMAND   PID USER   FD   TYPE             DEVICE SIZE/OFF       NODE NAME
・・・
rsyslogd 1159 root    2w   REG              252,3    88409       6993 /var/log/messages
rsyslogd 1159 root    3r   REG              252,3       41      12516 /var/log/local0.log
rsyslogd 1159 root    4w   REG              252,3    66124        545 /var/log/cron
```

  - 「/var/log/local0.log」をオープンしていることが分かります。

- mvコマンドで「/var/log/local0.log」をバックアップします。

```
[root@vm1-□ ~]# mv /var/log/local0.log /var/log/local0.log.bak
```

- rsyslogdがオープンしているファイルを再度、確認します。

```
[root@vm1-□ ~]# lsof -p 1159
COMMAND   PID USER   FD   TYPE             DEVICE SIZE/OFF       NODE NAME
・・・
rsyslogd 1159 root    2w   REG              252,3    88409       6993 /var/log/messages
rsyslogd 1159 root    3r   REG              252,3       41      12516 /var/log/local0.log.bak
rsyslogd 1159 root    4w   REG              252,3    66124        545 /var/log/cron
```

  - 「/var/log/local0.log.bak」をオープンしていることが分かります。オープン中のファイルはinode番号（「NODE」の値）で管理されており、mvコマンドはファイル名を変更するだけで、inode番号は変更しないためこのような動きになります。

システムログの収集 (4)

- 再度、Facilityがlocal0のログを出力すると、「/var/log/local0.log.bak」に記録されることを確認します。

```
[root@vm1-□ ~]# logger -p local0.info -t 'Test' 'Hello, World - Again!'
[root@vm1-□ ~]# cat /var/log/local0.log.bak
Jan 19 19:48:39 vm1-□ Test: Hello, World!
Jan 19 20:02:41 vm1-□ Test: Hello, World - Again!
```

- 新しいログファイルに記録するためにrsyslogサービスを再起動します。

```
[root@vm1-□ ~]# service rsyslog restart
システムロガーを停止中:                                    [  OK  ]
システムロガーを起動中:                                    [  OK  ]
```

  - 次のようにrsyslogdプロセスにHUPシグナルを送信しても構いません。

```
[root@vm1-□ ~]# pkill -HUP rsyslogd
```

- Facilityがlocal0のログを出力すると、「/var/log/local0.log」に記録されることを確認します。

```
[root@vm1-□ ~]# logger -p local0.info -t 'Test' 'Hello, World - One more!'
[root@vm1-□ ~]# cat /var/log/local0.log
Jan 19 20:04:51 vm1-□ Test: Hello, World - One more!
```

- 質問１の解答例は以上です。

システムログの収集 (5)

- 質問２: 次の設定方法を考えてください。
  - すべてのカーネルメッセージが「/var/log/kernel.log」 に記録されるように設定します。
  - VM1-□を再起動して、出力されたカーネルメッセージを確認してください。

システムログの収集 (6)

- 質問２の解答例
  - ホストLinuxのコマンド端末からVM1-□にログインします。

```
# ssh root@192.168.122.10□    ← ホストLinuxのコマンド端末で実行
```

  - 設定ファイル「/etc/rsyslog.conf」に次の設定を追加します。

/etc/rsyslog.conf

```
# Save boot messages also to boot.log
local7.*                                                /var/log/boot.log

local0.*                                                /var/log/local0.log

kern.*                                                  /var/log/kernel.log    ← この行を追加
```

  - rsyslogサービスをreloadして設定変更を反映します。

```
[root@vm1-□ ~]# service rsyslog reload
Reloading system logger...                                 [  OK  ]
```

  - VM1-□を再起動します。

```
[root@vm1-□ ~]# reboot
```

システムログの収集 (7)

- VM1-□の再起動が完了したら、ホストLinuxのコマンド端末からVM1-□にログインします。

```
# ssh root@192.168.122.10□    ← ホストLinuxのコマンド端末で実行
```

- 「/var/log/kernel.log」の内容を確認します。

サーバ起動時のログ

```
Jan 19 20:14:45 vm1-□ kernel: imklog 4.6.2, log source = /proc/kmsg started.
Jan 19 20:14:53 vm1-□ kernel: type=1305 audit(1326971693.429:3652): audit_pid=0 old=1134
auid=4294967295 ses=4294967295 subj=system_u:system_r:auditd_t:s0 res=1
Jan 19 20:14:53 vm1-□ kernel: type=1305 audit(1326971693.494:3653): audit_enabled=0 old=1
auid=4294967295 ses=4294967295 subj=system_u:system_r:auditctl_t:s0 res=1
Jan 19 20:14:53 vm1-□ kernel: Kernel logging (proc) stopped.
Jan 19 20:15:21 vm1-□ kernel: imklog 4.6.2, log source = /proc/kmsg started.
Jan 19 20:15:21 vm1-□ kernel: Initializing cgroup subsys cpuset
Jan 19 20:15:21 vm1-□ kernel: Initializing cgroup subsys cpu
Jan 19 20:15:21 vm1-□ kernel: Linux version 2.6.32-220.el6.x86_64
(mockbuild@c6b18n3.bsys.dev.centos.org) (gcc version 4.4.6 20110731 (Red Hat 4.4.6-3)
(GCC) ) #1 SMP Tue Dec 6 19:48:22 GMT 2011
Jan 19 20:15:21 vm1-□ kernel: Command line: ro root=UUID=20a3788f-3e87-49ce-beb4-
93091fb922dc rd_NO_LUKS  KEYBOARDTYPE=pc KEYTABLE=us rd_NO_MD quiet rhgb crashkernel=auto
LANG=ja_JP.UTF-8 rd_NO_LVM rd_NO_DM
Jan 19 20:15:21 vm1-□ kernel: KERNEL supported cpus:
Jan 19 20:15:21 vm1-□ kernel:  Intel GenuineIntel
Jan 19 20:15:21 vm1-□ kernel:  AMD AuthenticAMD
Jan 19 20:15:21 vm1-□ kernel:  Centaur CentaurHauls
Jan 19 20:15:21 vm1-□ kernel: Disabled fast string operations
. . .
```

  - この例では、「Jan 19 20:14:21」にrsyslogdが起動して、それまでにカーネルログバッファに蓄積されたカーネルログがまとめてログファイルに出力されています。

- 質問２の解答例は以上です。
- 以上で「システムログの収集」は完了です。

コマンドによる情報収集 (1)

- 仮想マシン#1（VM1-□）において、コマンドによる情報収集を行います。
- 質問: 次の設定方法を考えてください。
  - 使用中のカーネルのバージョン (uname)
  - メモリの総容量、使用容量、空き容量 (free)
  - スワップ領域の総容量、使用容量、空き容量 (free)
  - ルートファイルシステムの総容量、使用容量、空き容量 (df)
  - サーバのIPアドレス (ifconfig)
  - サーバが接続を受け付ける(LISTEN 状態の) TCP/IP ポート (netstat)
  - 上記のTCP/IP ポートの接続を受け付けるプロセス (lsof)
  - サーバ上で稼動中のプロセスの個数 (ps)
  - ファイル /var/log/messages をオープンしているプロセス (lsof)
  - 導入済みのrpmパッケージの一覧 (rpm)
  - rpmパッケージcronie-anacronに含まれるファイル (rpm)
  - ファイル/etc/inittabと同じrpmパッケージに含まれるファイル (rpm)
  - 仮想マシンに接続された仮想PCIデバイス (lspci)
  - コマンドの詳細はmanページを確認してください。

コマンドによる情報収集 (2)

- 質問の解答例
  - ホストLinuxのコマンド端末からVM1-□にログインします。

```
# ssh root@192.168.122.10□    ← ホストLinuxのコマンド端末で実行
```

  - 使用中のカーネルのバージョン (uname)

```
[root@vm1-□ ~]# uname -a
Linux vm1-□ 2.6.32-220.el6.x86_64 #1 SMP Tue Dec 6 19:48:22 GMT 2011 x86_64 x86_64 x86_64 GNU/Linux
```

  - メモリの総容量、使用容量、空き容量 (free)

```
[root@vm1-□ ~]# free
             total       used       free     shared    buffers     cached
Mem:       1020692     209884     810808          0      11196      89864
-/+ buffers/cache:     108824     911868
Swap:       524280          0     524280
```

  - スワップ領域の総容量、使用容量、空き容量 (free)

```
[root@vm1-□ ~]# free
             total       used       free     shared    buffers     cached
Mem:       1020692     209884     810808          0      11196      89864
-/+ buffers/cache:     108824     911868
Swap:       524280          0     524280
```

コマンドによる情報収集 (3)

- ルートファイルシステムの総容量、使用容量、空き容量 (df)

```
[root@vm1-□ ~]# df
Filesystem           1K-ブロック      使用      使用可 使用% マウント位置
/dev/vda3              7538064   2337360   4817780  33% /
tmpfs                   510344         0    510344   0% /dev/shm
/dev/vda1               198337     27747    160350  15% /boot
```

- サーバのIPアドレス (ifconfig)

```
[root@vm1-□ ~]# ifconfig
eth0      Link encap:Ethernet  HWaddr 52:54:00:47:02:46
          inet addr:192.168.122.10□  Bcast:192.168.122.255  Mask:255.255.255.0
          inet6 addr: fe80::5054:ff:fe47:246/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1560 errors:0 dropped:0 overruns:0 frame:0
          TX packets:680 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:112166 (109.5 KiB)  TX bytes:200237 (195.5 KiB)
. . .
```

コマンドによる情報収集 (4)

- サーバが接続を受け付ける(LISTEN 状態の) TCP/IP ポート (netstat)

```
[root@vm1-□ ~]# netstat -nlt
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address               Foreign Address             State
tcp        0      0 0.0.0.0:111                 0.0.0.0:*                   LISTEN
tcp        0      0 0.0.0.0:22                  0.0.0.0:*                   LISTEN
tcp        0      0 127.0.0.1:631               0.0.0.0:*                   LISTEN
tcp        0      0 127.0.0.1:25                0.0.0.0:*                   LISTEN
tcp        0      0 0.0.0.0:3260                0.0.0.0:*                   LISTEN
tcp        0      0 0.0.0.0:5672                0.0.0.0:*                   LISTEN
tcp        0      0 0.0.0.0:52489               0.0.0.0:*                   LISTEN
. . .
```

- 上記のTCP/IP ポートの接続を受け付けるプロセス (lsof)

```
[root@vm1-□ ~]# lsof -iTCP -sTCP:LISTEN
COMMAND    PID    USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
rpcbind   1216     rpc    8u  IPv4  11589      0t0  TCP *:sunrpc (LISTEN)
rpcbind   1216     rpc   11u  IPv6  11594      0t0  TCP *:sunrpc (LISTEN)
rpc.statd 1234 rpcuser    9u  IPv4  11689      0t0  TCP *:52489 (LISTEN)
rpc.statd 1234 rpcuser   11u  IPv6  11697      0t0  TCP *:60115 (LISTEN)
cupsd     1309    root    6u  IPv6  12006      0t0  TCP localhost:ipp (LISTEN)
cupsd     1309    root    7u  IPv4  12007      0t0  TCP localhost:ipp (LISTEN)
tgtd      1423    root    4u  IPv4  12545      0t0  TCP *:iscsi-target (LISTEN)
tgtd      1423    root    5u  IPv6  12546      0t0  TCP *:iscsi-target (LISTEN)
tgtd      1425    root    4u  IPv4  12545      0t0  TCP *:iscsi-target (LISTEN)
tgtd      1425    root    5u  IPv6  12546      0t0  TCP *:iscsi-target (LISTEN)
sshd      1455    root    3u  IPv4  12657      0t0  TCP *:ssh (LISTEN)
sshd      1455    root    4u  IPv6  12661      0t0  TCP *:ssh (LISTEN)
master    1531    root   12u  IPv4  12854      0t0  TCP localhost:smtp (LISTEN)
master    1531    root   13u  IPv6  12856      0t0  TCP localhost:smtp (LISTEN)
httpd     1571    root    4u  IPv6  13111      0t0  TCP *:http (LISTEN)
httpd     1577  apache    4u  IPv6  13111      0t0  TCP *:http (LISTEN)
. . .
```

コマンドによる情報収集 (5)

- サーバ上で稼動中のプロセスの個数 (ps)

```
[root@vm1-□ ~]# ps -ef | wc -l
113
```

- ファイル /var/log/messages をオープンしているプロセス (lsof)

```
[root@vm1-□ ~]# lsof /var/log/messages
COMMAND    PID USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
rsyslogd  1173 root    4w   REG  252,3   118750 6993 /var/log/messages
abrt-dump 1563 root    4r   REG  252,3   118750 6993 /var/log/messages
```

- 導入済みのrpmパッケージの一覧 (rpm)

```
[root@vm1-□ ~]# rpm -qa
acpid-1.0.10-2.1.el6.x86_64
libsndfile-1.0.20-5.el6.x86_64
tzdata-2011l-4.el6.noarch
cpuspeed-1.5-15.el6.x86_64
poppler-utils-0.12.4-3.el6_0.1.x86_64
foomatic-db-filesystem-4.0-7.20091126.el6.noarch
prelink-0.4.6-3.el6.x86_64
mysql-libs-5.1.52-1.el6_0.1.x86_64
iso-codes-3.16-2.el6.noarch
latencytop-0.5-3.el6.x86_64
krb5-workstation-1.9-22.el6.x86_64
ncurses-base-5.7-3.20090208.el6.x86_64
cas-0.15-1.el6.1.noarch
ConsoleKit-libs-0.4.1-3.el6.x86_64
ncurses-libs-5.7-3.20090208.el6.x86_64
grub-0.97-75.el6.x86_64
nss-3.12.10-16.el6.x86_64
zlib-1.2.3-27.el6.x86_64
vim-enhanced-7.2.411-1.6.el6.x86_64
matahari-lib-0.4.4-11.el6.x86_64
. . .
```

コマンドによる情報収集 (6)

- rpmパッケージcronie-anacronに含まれるファイル (rpm)

```
[root@vm1-□ ~]# rpm -ql cronie-anacron
/etc/anacrontab
/etc/cron.hourly/0anacron
/usr/sbin/anacron
/usr/share/man/man5/anacrontab.5.gz
/usr/share/man/man8/anacron.8.gz
/var/spool/anacron
/var/spool/anacron/cron.daily
/var/spool/anacron/cron.monthly
/var/spool/anacron/cron.weekly
```

- ファイル/etc/inittabと同じrpmパッケージに含まれるファイル (rpm)

```
[root@vm1-□ ~]# rpm -qf /etc/inittab
initscripts-9.03.27-1.el6.centos.x86_64

[root@vm1-□ ~]# rpm -ql initscripts
/bin/ipcalc
/bin/usleep
/etc/NetworkManager
/etc/NetworkManager/dispatcher.d
/etc/NetworkManager/dispatcher.d/00-netreport
/etc/NetworkManager/dispatcher.d/05-netfs
/etc/X11/prefdm
/etc/adjtime
/etc/init/control-alt-delete.conf
/etc/init/kexec-disable.conf
/etc/init/plymouth-shutdown.conf
/etc/init/prefdm.conf
/etc/init/quit-plymouth.conf
/etc/init/rc.conf
. . .
```

コマンドによる情報収集 (7)

- 仮想マシンに接続された仮想PCIデバイス (lspci)

```
[root@vm1-□ ~]# lspci
00:00.0 Host bridge: Intel Corporation 440FX - 82441FX PMC [Natoma] (rev 02)
00:01.0 ISA bridge: Intel Corporation 82371SB PIIX3 ISA [Natoma/Triton II]
00:01.1 IDE interface: Intel Corporation 82371SB PIIX3 IDE [Natoma/Triton II]
00:01.2 USB controller: Intel Corporation 82371SB PIIX3 USB [Natoma/Triton II] (rev 01)
00:01.3 Bridge: Intel Corporation 82371AB/EB/MB PIIX4 ACPI (rev 03)
00:02.0 VGA compatible controller: Cirrus Logic GD 5446
00:03.0 Ethernet controller: Red Hat, Inc Virtio network device
00:04.0 SCSI storage controller: Red Hat, Inc Virtio block device
00:05.0 RAM memory: Red Hat, Inc Virtio memory balloon
```

- 質問の解答例は以上です。
- 以上で「コマンドによる情報収集」は完了です。

- 以上で「問題判別基本コマンド演習」は完了です。

Upstartジョブ管理演習

演習内容

- この演習では、次の作業を行います。
  - 仮想マシン#1（VM1-□）において、Upstartの動作の仕組みを確認します。
    - 独自のUpstartジョブを定義して実行します。
    - TTYコンソールのジョブにおける「respawn」の動作を確認します。（TTYコンソールは、virt-managerから使用する仮想コンソールのことです。）
    - 「respawn」指定のジョブが障害で繰り返し停止する場合の動作を確認します。

Upstartジョブの作成 (1)

- 仮想マシン#1（VM1-□）において、システムログに日付を出力するジョブを作成して、Upstartの動作の仕組みを確認します。
  - ホストLinuxのコマンド端末からVM1-□にログインします。

```
# ssh root@192.168.122.10□    ← ホストLinuxのコマンド端末で実行
```

  - シェルスクリプト「/root/work/logdate.sh」を次の内容で作成します。

/root/work/logdate.sh

```
#!/bin/sh

while [[ true ]]; do
    logger -t "UpstartJob" $(date)
    sleep 1
done
```

  - シェルスクリプトに実行権を設定します。

```
# chmod u+x /root/work/logdate.sh
```

  - 手動でシェルスクリプトを実行して動作確認をします。

```
[root@vm1-□ ~]# /root/work/logdate.sh
^C                                                  ← 数秒待ってから、Ctrl+Cで停止
[root@vm1-□ ~]# tail /var/log/messages
. . .
Jan 19 21:23:15 vm1-□ UpstartJob: 2012年 1月 19日 木曜日 21:23:15 JST
Jan 19 21:23:16 vm1-□ UpstartJob: 2012年 1月 19日 木曜日 21:23:16 JST
Jan 19 21:23:17 vm1-□ UpstartJob: 2012年 1月 19日 木曜日 21:23:17 JST
Jan 19 21:23:18 vm1-□ UpstartJob: 2012年 1月 19日 木曜日 21:23:18 JST
```

Upstartジョブの作成 (2)

- ジョブ定義ファイル「/etc/init/logdate.conf」を次の内容で作成します。

/etc/init/logdate.conf

```
start on logdate_run
stop on logdate_stop

respawn
exec /root/work/logdate.sh
```

  - イベント「logdate_run」で/root/work/logdate.shを実行して、イベント「logdate_stop」で停止するという定義です。「respawn」は、ジョブのプロセスが終了した際に自動的に再起動する指定です。

Upstartジョブの作成 (3)

- initctlコマンドで定義済みのジョブを表示して、logdateが存在することを確認します。

```
[root@vm1-□ ~]# initctl list
rc stop/waiting
tty (/dev/tty3) start/running, process 1653
tty (/dev/tty2) start/running, process 1651
tty (/dev/tty1) start/running, process 1649
tty (/dev/tty6) start/running, process 1659
tty (/dev/tty5) start/running, process 1657
tty (/dev/tty4) start/running, process 1655
plymouth-shutdown stop/waiting
control-alt-delete stop/waiting
rcS-emergency stop/waiting
readahead-collector stop/waiting
kexec-disable stop/waiting
quit-plymouth stop/waiting
rcS stop/waiting
prefdm stop/waiting
init-system-dbus stop/waiting
readahead stop/waiting
splash-manager stop/waiting
start-ttys stop/waiting
readahead-disable-services stop/waiting
logdate stop/waiting
rcS-sulogin stop/waiting
serial stop/waiting
```

- 新しいコマンド端末からVM1-□にログインして、「/var/log/messages」の出力を観察しておきます。

```
# tail -f /var/log/messages
```

Upstartジョブの作成 (4)

- initctlコマンドでイベント「logdate_run」を発生して、logdateジョブの実行が開始されることを確認します。

```
[root@vm1-□ ~]# initctl emit "logdate_run"
[root@vm1-□ ~]# initctl list | grep logdate
logdate start/running, process 2035
```

  - システムログ「/var/log/messages」に1秒ごとに日付が記録されます。

- ジョブのプロセスlogdate.shを強制停止します。この時、respawn指定により、再度、ジョブが開始することを確認します。

```
[root@vm1-□ ~]# pkill "logdate\.sh"
[root@vm1-□ ~]# initctl list | grep logdate
logdate start/running, process 2112
```

- initctlコマンドでイベント「logdate_stop」を発生して、logdateジョブが終了することを確認します。

```
[root@vm1-□ ~]# initctl emit "logdate_stop"
[root@vm1-□ ~]# initctl list | grep logdate
logdate stop/waiting
```

- 次のコマンドでもlogdateジョブの開始、終了が行えます。
  - # initctl start logdate
  - # initctl stop logdate
- 以上で「Upstartジョブの作成」は完了です。

TTYコンソールジョブの動作確認 (1)

- TTYコンソールのジョブにおけるrespawn動作の確認を行います。
  - ホストLinuxのコマンド端末からVM1-□にログインします。

```
# ssh root@192.168.122.10□    ← ホストLinuxのコマンド端末で実行
```

  - TTYコンソールのジョブが実行中であることを確認します。

```
[root@vm1-□ ~]# initctl list | grep tty
tty (/dev/tty3) start/running, process 4207
tty (/dev/tty2) start/running, process 4205
tty (/dev/tty1) start/running, process 4203
tty (/dev/tty6) start/running, process 4213
tty (/dev/tty5) start/running, process 4211
tty (/dev/tty4) start/running, process 4209
start-ttys stop/waiting
```

  - 設定ファイル「/etc/init/tty.conf」をコピーしてバックアップします。

```
# cp /etc/init/tty.conf /root/work/tty.conf
```

  - 「/etc/init/tty.conf」を編集して「resqpwn」指定をコメントアウトします。

/etc/init/tty.conf

```
# tty - getty
#
# This service maintains a getty on the specified device.

stop on runlevel [S016]

#respawn                          ←この行をコメントアウト
instance $TTY
exec /sbin/mingetty $TTY
```

TTYコンソールジョブの動作確認 (2)

- 次のコマンドでTTYコンソールのジョブを再起動します。

```
[root@vm1-□ ~]# for i in $(seq 1 6); do initctl stop tty TTY=/dev/tty$i; done
tty stop/waiting
tty stop/waiting
tty stop/waiting
tty stop/waiting
tty stop/waiting
tty stop/waiting

[root@vm1-□ ~]# initctl list | grep tty
tty stop/waiting
start-ttys stop/waiting

[root@vm1-□ ~]# initctl start start-ttys RUNLEVEL=3
start-ttys stop/waiting

[root@vm1-□ ~]# initctl list | grep tty
tty (/dev/tty3) start/running, process 4275
tty (/dev/tty2) start/running, process 4273
tty (/dev/tty1) start/running, process 4271
tty (/dev/tty6) start/running, process 4281
tty (/dev/tty5) start/running, process 4279
tty (/dev/tty4) start/running, process 4277
start-ttys stop/waiting
```

- 「start-ttys」はttyジョブを開始するためのラッパーです。最後に「start-tty stop/waiting」と表示されるのは、ラッパーのジョブが終了したという意味です。

TTYコンソールジョブの動作確認 (3)

- ホストLinuxでvirt-managerを起動します。

```
# virt-manager
```

- 「VM1-□」をダブルクリックして仮想コンソールを開き、rootユーザでログインして、すぐにログアウトします。

```
CentOS release 6.2 (Final)
Kernel 2.6.32-220.el6.x86_64 on an x86_64

vm1-1 login: root
Password:
Last login: Thu May 17 10:01:03 on tty1
[root@vm1-1 ~]# exit
logout
_
```

  - TTYコンソールジョブに「respawn」指定がないため、ログアウトすると、ログインプロンプトが再表示されません。

- コマンド端末の方で、ttyジョブの状態を確認します。

```
[root@vm1-□ ~]# initctl list | grep tty
tty (/dev/tty3) start/running, process 4275
tty (/dev/tty2) start/running, process 4273
tty (/dev/tty6) start/running, process 4281
tty (/dev/tty5) start/running, process 4279
tty (/dev/tty4) start/running, process 4277
start-ttys stop/waiting
```

  - 「/dev/tty1」のttyジョブが終了して、存在しなくなっています。

TTYコンソールジョブの動作確認 (4)

- 「/etc/init/tty.conf」のrespawn指定を元にもどします。

/etc/init/tty.conf

```
# tty - getty
#
# This service maintains a getty on the specified device.

stop on runlevel [S016]

respawn                         ←この行を修正
instance $TTY
exec /sbin/mingetty $TTY
```

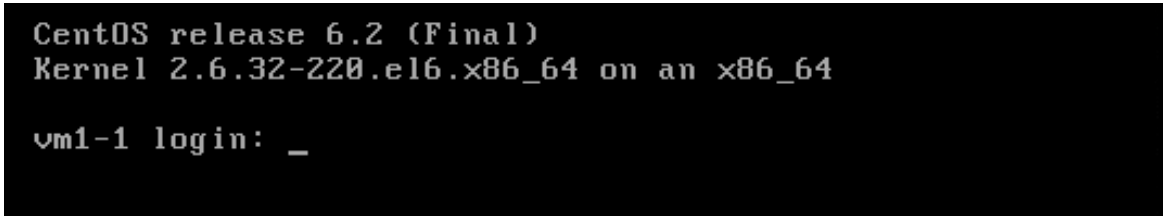
- 次のコマンドでTTYコンソールのジョブを再起動します。

```
[root@vm1-□ ~]# for i in $(seq 1 6); do initctl stop tty TTY=/dev/tty$i; done
initctl: Unknown instance: /dev/tty1
tty stop/waiting
tty stop/waiting
tty stop/waiting
tty stop/waiting
tty stop/waiting

[root@vm1-□ ~]#  initctl start start-ttys RUNLEVEL=3
start-ttys stop/waiting

[root@vm1-□ ~]#  initctl list | grep tty
tty (/dev/tty3) start/running, process 4435
tty (/dev/tty2) start/running, process 4433
tty (/dev/tty1) start/running, process 4431
tty (/dev/tty6) start/running, process 4441
tty (/dev/tty5) start/running, process 4439
tty (/dev/tty4) start/running, process 4437
start-ttys stop/waiting
```

- 「initctl: Unknown instance: /dev/tty1」の表示は、「/dev/tty1」のプロセスが存在しないためです。

TTYコンソールジョブの動作確認 (5)

- 仮想コンソールにログインプロンプトが再表示されています。ログイン、ログアウトを行なって、ログインプロンプトが再表示されることを確認します。



- 以上で「TTYコンソールジョブの動作確認」は完了です。

ジョブ障害時の動作確認 (1)

- ジョブが障害で繰り返し停止した場合の動作を確認します。
  - ホストLinuxのコマンド端末からVM1-□にログインします。

```
# ssh root@192.168.122.10□    ← ホストLinuxのコマンド端末で実行
```

  - 先の演習で作成したシェルスクリプト「/root/work/logdate.sh」を次のように修正します。

/root/work/logdate.sh

```
#!/bin/sh

while [[ true ]]; do
    logger -t "UpstartJob" $(date)
    exit                                ← この行を追加
    sleep 1
done
```

    - 一度だけ日付を出力してすぐに停止するようになります。

  - システムログの出力を観察します。

```
# tail -f /var/log/messages
```

  - 新しいコマンド端末からVM1-□にログインして、logdateジョブを開始します。

```
[root@vm1-□ ~]# initctl emit "logdate_run"
```

ジョブ障害時の動作確認 (2)

- システムログの出力から、連続してジョブが再起動された後に、ジョブの再起動が中止されていることを確認します。

```
Jan 20 12:06:52 vm1-□ UpstartJob: Fri Jan 20 12:06:52 JST 2012
Jan 20 12:06:52 vm1-□ init: logdate main process ended, respawning
Jan 20 12:06:52 vm1-□ UpstartJob: Fri Jan 20 12:06:52 JST 2012
Jan 20 12:06:52 vm1-□ init: logdate main process ended, respawning
Jan 20 12:06:52 vm1-□ UpstartJob: Fri Jan 20 12:06:52 JST 2012
Jan 20 12:06:52 vm1-□ init: logdate main process ended, respawning
Jan 20 12:06:52 vm1-□ UpstartJob: Fri Jan 20 12:06:52 JST 2012
Jan 20 12:06:52 vm1-□ init: logdate main process ended, respawning
Jan 20 12:06:52 vm1-□ UpstartJob: Fri Jan 20 12:06:52 JST 2012
Jan 20 12:06:52 vm1-□ init: logdate main process ended, respawning
Jan 20 12:06:52 vm1-□ UpstartJob: Fri Jan 20 12:06:52 JST 2012
Jan 20 12:06:52 vm1-□ init: logdate main process ended, respawning
Jan 20 12:06:52 vm1-□ UpstartJob: Fri Jan 20 12:06:52 JST 2012
Jan 20 12:06:52 vm1-□ init: logdate main process ended, respawning
Jan 20 12:06:52 vm1-□ UpstartJob: Fri Jan 20 12:06:52 JST 2012
Jan 20 12:06:52 vm1-□ init: logdate main process ended, respawning
Jan 20 12:06:52 vm1-□ UpstartJob: Fri Jan 20 12:06:52 JST 2012
Jan 20 12:06:52 vm1-□ init: logdate main process ended, respawning
Jan 20 12:06:52 vm1-□ UpstartJob: Fri Jan 20 12:06:52 JST 2012
Jan 20 12:06:52 vm1-□ init: logdate main process ended, respawning
Jan 20 12:06:52 vm1-□ UpstartJob: Fri Jan 20 12:06:52 JST 2012
Jan 20 12:06:52 vm1-□ init: logdate respawning too fast, stopped
```

  - 最後の「respawning too fast, stopped」というメッセージから、respawn指定のジョブが障害で何度も停止したことが分かります。

- logdateジョブが停止していることを確認します。

```
[root@vm1-□ ~]# initctl list | grep logdate
logdate stop/waiting
```


NII National Institute of Informatics

ジョブ障害時の動作確認 (3)

- 以上で「ジョブ障害時」の動作確認は完了です。

- 以上で「Upstartジョブ管理演習」は完了です。

メモとしてお使いください

メモとしてお使いください

プロセス管理/メモリ管理演習

演習内容

- この演習では、次の作業を行います。
  - 仮想マシン#1（VM1-□）において、CPU、メモリなどのサーバリソースをプロセス使用する状況を確認します。
    - tmpfsがメモリを使用する状況を確認します。
    - vmstatコマンド、mpstatコマンド、topコマンドで、CPU使用状況を確認します。
    - メモリを大量に消費するプロセスにより、スワップアウトが発生する状況を確認します。
    - メモリを大量に消費するプロセスにより、OOM Killerが発生する状況を確認します。

tmpfsによるメモリ使用の確認 (1)

- 仮想マシン#1（VM1-□）において、tmpfsがメモリを使用する状況を確認します。
  - ホストLinuxのコマンド端末からVM1-□にログインします。

```
# ssh root@192.168.122.10□    ← ホストLinuxのコマンド端末で実行
```

  - ディスクキャッシュを解放します。

```
[root@vm1-□ ~]# echo 3 > /proc/sys/vm/drop_caches
```

  - tmpfsを利用して、512MBのラムディスクを「/mnt/ramdisk」にマウントします。

```
[root@vm1-□ ~]# mkdir /mnt/ramdisk
[root@vm1-□ ~]# mount -t tmpfs -o size=512M tmpfs /mnt/ramdisk
```

  - 500MBのファイルをラムディスクに作成します。

```
[root@vm1-□ ~]# dd if=/dev/zero of=/mnt/ramdisk/test bs=1M count=500
500+0 records in
500+0 records out
524288000 bytes (524 MB) copied, 0.889495 s, 589 MB/s
```

  - 「/mnt/ramdisk」の使用容量を確認します。

```
[root@vm1-□ ~]# df
Filesystem           1K-ブロック      使用      使用可 使用% マウント位置
/dev/vda3              7538064   2338196   4816944  33% /
tmpfs                   510344         0    510344   0% /dev/shm
/dev/vda1               198337     27747    160350  15% /boot
tmpfs                   524288    513004     11284  98% /mnt/ramdisk
```

tmpfsによるメモリ使用の確認 (2)

- ディスクキャッシュの使用量を確認します。

```
[root@vm1-□ ~]# free
             total       used       free     shared    buffers     cached
Mem:       1020692     643632     377060          0       1860     528152
-/+ buffers/cache:     113620     907072
Swap:       524280          0     524280
```

  - tmpfsの使用容量に対応して、約500MBのディスクキャッシュが使用されています。

- ディスクキャッシュを解放してもディスクキャッシュの使用量は500MB以下にならないことを確認します。

```
[root@vm1-□ ~]# echo 3 > /proc/sys/vm/drop_caches
[root@vm1-□ ~]# free
             total       used       free     shared    buffers     cached
Mem:       1020692     640168     380524          0        212     526540
-/+ buffers/cache:     113416     907276
Swap:       524280          0     524280
```

- tmpfsをアンマウントすると、ディスクキャッシュの使用量を確認します。

```
[root@vm1-□ ~]# umount /mnt/ramdisk
[root@vm1-□ ~]# free
             total       used       free     shared    buffers     cached
Mem:       1020692     129040     891652          0       1536      15728
-/+ buffers/cache:     111776     908916
Swap:       524280          0     524280
```

  - ディスクキャッシュの使用量が500MB程度、減少しています。

- 以上で「tmpfsによるメモリ使用の確認」は完了です。

CPU使用状況の確認 (1)

- 仮想マシン#1（VM1-□）において、各種コマンドによるCPU使用状況の見え方を確認します。
  - ホストLinuxのコマンド端末からVM1-□にログインします。

```
# ssh root@192.168.122.10□    ← ホストLinuxのコマンド端末で実行
```

  - CPUを使用しつづけるプロセスを実行します。

```
[root@vm1-□ ~]# cat /dev/zero > /dev/null
```

  - 新しいコマンド端末からVM1-□にログインして、vmstatコマンドでCPU使用率を確認します。

```
[root@vm1-□ ~]# vmstat 1
procs -----------memory---------- ---swap-- -----io---- --system-- -----cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 1  0      0 824568   5448  78556    0    0     2     0   13   10  0  0 100  0  0
 1  0      0 824568   5448  78556    0    0     0     0 1035   26  2 48 50  0  0
 1  0      0 824560   5448  78556    0    0     0     0 1019   23  1 49 50  0  0
 1  0      0 824560   5448  78556    0    0     0     0 1020   19  2 49 50  0  0
 1  0      0 824560   5448  78556    0    0     0     0 1019   20  2 49 50  0  0
^C                                                                ← Ctrl+Cで停止
```

    - CPU使用率が約50%であることが分かります。次で確認するように、2個の仮想CPUに対して、1個が100%使用されているため、全仮想CPUの「平均使用率」が50%となっています。

CPU使用状況の確認 (2)

- mpstatコマンドで、仮想CPUごとの使用率を確認します。

```
[root@vm1-□ ~]# mpstat -P ALL 1
Linux 2.6.32-220.el6.x86_64 (vm1-□)       2012年01月20日     _x86_64_    (2 CPU)

14時30分39秒  CPU    %usr   %nice    %sys %iowait    %irq   %soft  %steal  %guest   %idle
14時30分40秒  all    1.01    0.00   48.74    0.00    0.00    0.00    0.00    0.00   50.25
14時30分40秒    0    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00  100.00
14時30分40秒    1    2.00    0.00   98.00    0.00    0.00    0.00    0.00    0.00    0.00

14時30分40秒  CPU    %usr   %nice    %sys %iowait    %irq   %soft  %steal  %guest   %idle
14時30分41秒  all    1.50    0.00   48.50    1.00    0.00    0.00    0.00    0.00   49.00
14時30分41秒    0    0.00    0.00    0.00    2.00    0.00    0.00    0.00    0.00   98.00
14時30分41秒    1    3.96    0.00   96.04    0.00    0.00    0.00    0.00    0.00    0.00
^C                                                                    ← Ctrl+Cで停止
```

  - 仮想CPU#1がほぼ100%使用されていることが分かります。

- topコマンドでCPUを使用しているプロセスを確認します。

```
[root@vm1-□ ~]# top

top - 14:33:35 up 18:18,  3 users,  load average: 0.99, 0.75, 0.35
Tasks: 117 total,   2 running, 115 sleeping,   0 stopped,   0 zombie
Cpu0  :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu1  :  2.7%us, 97.3%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   1020692k total,   197124k used,   823568k free,     5652k buffers
Swap:   524280k total,        0k used,   524280k free,    78696k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 5550 root      20   0 98.6m  576  472 R 100.0  0.1   6:58.32 cat
    1 root      20   0 19400 1588 1256 S  0.0  0.2   0:00.73 init
    2 root      20   0     0    0    0 S  0.0  0.0   0:00.00 kthreadd
```

  - topコマンド実行中に「1」を押すとCPUごとの使用率が表示されます。「q」で終了します。

CPU使用状況の確認 (3)

- さらに新しいコマンド端末からVM1-□にログインして、CPUを使用し続けるプロセスをもう1つ実行します。

```
[root@vm1-□ ~]# cat /dev/zero > /dev/null
```

- topコマンドでCPUの使用状況を確認します。

```
[root@vm1-□ ~]# top

top - 14:39:49 up 18:24,  3 users,  load average: 1.22, 0.97, 0.57
Tasks: 118 total,   3 running, 115 sleeping,   0 stopped,   0 zombie
Cpu0  :  3.0%us, 97.0%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu1  :  2.3%us, 97.7%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   1020692k total,   198000k used,   822692k free,     5708k buffers
Swap:   524280k total,        0k used,   524280k free,    79236k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 5550 root      20   0 98.6m  576  472 R 100.0  0.1  13:09.65 cat
 5582 root      20   0 98.6m  580  472 R 99.8  0.1   0:12.59 cat
    1 root      20   0 19400 1588 1256 S  0.0  0.2   0:00.73 init
```

  - 2個のcatプロセスがそれぞれCPU#0とCPU#1を専有して使用していることが分かります。

- 以上で「CPU使用状況の確認」は完了です。
  - 実行中のcatコマンドは「Ctrl+C」で停止しておきます。

スワップアウト発生状況の確認 (1)

- 仮想マシン#1（VM1-□）において、スワップアウト発生の状況を確認します。
  - ホストLinuxのコマンド端末からVM1-□にログインします。

```
# ssh root@192.168.122.10□    ← ホストLinuxのコマンド端末で実行
```

  - 指定量のメモリを消費するPerlスクリプト「/root/work/mleak.pl」を次の内容で作成します。

/root/work/mleak.pl

```
#!/usr/bin/perl

$c = shift @ARGV;
print "Consuming $c MB Memory...\n";
for ( $i = 0; $i < $c; $i++ ) {
      $a[$i] = "Z" x 1024 x 1024;
}
print "Done. Hit Enter to exit.\n";
$_=<>;
```

    - 作成したスクリプトに実行権を設定します。

```
# chmod u+x /root/work/mleak.pl
```

  - vmstatコマンドでメモリとスワップ領域の使用状況を監視します。

```
[root@vm1-□ ~]# vmstat 1
procs -----------memory---------- ---swap-- -----io---- --system-- -----cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 0  0      0 819244   6784  82240    0    0     2     0   37   10  0  1 99  0  0
 0  0      0 819236   6784  82240    0    0     0     0   35   17  0  0 100  0  0
 0  0      0 819236   6784  82240    0    0     0     0   33   21  0  0 100  0  0
```

    - vmstatコマンドはこのまま実行を続けます。

スワップアウト発生状況の確認 (2)

- 新しいコマンド端末からVM1-□にログインして、先に用意したスクリプトで1024MBのメモリを消費します。

```
[root@vm1-□ work]# cd /root/work
[root@vm1-□ work]# ./mleak.pl 1024
Consuming 1024 MB Memory...
Done. Hit Enter to exit.
```

- vmstatの出力からスワップアウトが発生していることを確認します。

```
procs -----------memory---------- ---swap-- -----io---- --system-- -----cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 1  0      0 893600   1408  13408    0    0     0     0   35   20  0  1 99  0  0
 1  0      0 446008   1408  13556    0    0   148     0  545   64 14  6 78  1  0
 1  2 162520  49012   1408  12548  200 166768   200 166780 33350  267 12 22 35 31  0
 0  2 271408  49088   1368  11984  156 104768   156 104768 11995  406  5 15 27 53  0
 0  0 277008  72796    468   5120  128 5652   276  5652  182  256  0  0 70 30  0
 0  0 277008  72796    468   5120    0    0     0     0   25   19  0  0 100  0  0
 0  0 277008  72796    468   5120    0    0     0     0   27   23  0  0 100  0  0
 1  0 277008  72796    468   5120    0    0     0     0   25   15  0  0 100  0  0
```

- さらに新しいコマンド端末からVM1-□にログインして、メモリ使用量を確認します。

```
[root@vm1-□ ~]# free
             total       used       free     shared    buffers     cached
Mem:       1020692     948904      71788          0        468       5272
-/+ buffers/cache:     943164      77528
Swap:       524280     276852     247428
```

  - メモリの空き容量が少なく、スワップ領域が使用されていることが分かります。

スワップアウト発生状況の確認 (3)

- 「mleak.pl」を実行したコマンド端末で[Enter]を押して、スクリプトを終了します。
- スワップ領域を無効化して、再度、有効化します。

```
[root@vm1-□ ~]# swapoff -a
[root@vm1-□ ~]# swapon -a
```

- vmstatの出力からスワップインが発生していることを確認します。

```
procs -----------memory---------- ---swap-- -----io---- --system-- -----cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 1  0  24976 921520   1580   7196    0    0     0     0   25   15  0  0 100  0  0
 0  0  24976 921520   1580   7196    0    0     0     0   34   24  0  0 100  0  0
 0  0  24916 921396   1580   7196  152    0   152     0   48   46  0  0 100  0  0
 0  0  24916 921396   1580   7196    0    0     0     0   24   17  0  0 100  0  0
 2  1   8728 907192   1580   7496 16252    0 16252     0 4641 12175  0 10 70 21  0
 0  0      0 899876   1576   7508 8232    0  8872     0 2704 6802  0  6 76 17  0
 0  0      0 899876   1576   7508    0    0     0     0   41   27  0  0 100  0  0
 0  0      0 899876   1576   7508    0    0     0     0   25   24  0  0 100  0  0
```

- スワップ領域の使用量が0になっていることを確認します。

```
[root@vm1-□ ~]# free
             total       used       free     shared    buffers     cached
Mem:       1020692     120916     899776          0       1596       7508
-/+ buffers/cache:     111812     908880
Swap:       524280          0     524280
```

- 以上で「スワップアウト発生状況の確認」は完了です。

OOM Killer発生状況の確認 (1)

- 仮想マシン#1（VM1-□）において、OOM Killer発生の状況を確認します。
  - ホストLinuxのコマンド端末からVM1-□にログインします。

```
# ssh root@192.168.122.10□    ← ホストLinuxのコマンド端末で実行
```

  - スワップ領域の使用量を監視します。

```
[root@vm1-□ ~]# watch -n1 free
```

  - 新しいコマンド端末からVM1-□にログインして、先の演習で作成したスクリプト「mleak.pl」で、2048MBのメモリを消費します。

```
[root@vm1-□ ~]# cd /root/work/
[root@vm1-□ work]# ./mleak.pl 2048
Consuming 2048 MB Memory...
強制終了
```

    - 「物理メモリ + スワップ容量」を超えるメモリが消費されるため、OOM Killerが発生して、プロセスが強制停止されます。
    - スワップ領域の使用量が増加していき、スワップ領域の残りが少なくなった所で、強制終了が発生することが分かります。

```
             total       used       free     shared    buffers     cached
Mem:       1020692     964808      55884          0        600       8536
-/+ buffers/cache:     955672      65020
Swap:       524280     436120      88160
```

OOM Killer発生状況の確認 (2)

- システムログ「/var/log/messages」を開いて、OOM Killer発生時のログが記録されていることを確認します。

```
Jan 20 15:45:16 vm1-□ kernel: mleak.pl invoked oom-killer: gfp_mask=0x280da, order=0, oom_adj=0,
oom_score_adj=0
Jan 20 15:45:16 vm1-□ kernel: mleak.pl cpuset=/ mems_allowed=0
Jan 20 15:45:16 vm1-□ kernel: Pid: 5963, comm: mleak.pl Not tainted 2.6.32-220.el6.x86_64 #1
Jan 20 15:45:16 vm1-□ kernel: Call Trace:
Jan 20 15:45:16 vm1-□ kernel: [<ffffffff810c2cb1>] ? cpuset_print_task_mems_allowed+0x91/0xb0
Jan 20 15:45:16 vm1-□ kernel: [<ffffffff81113a30>] ? dump_header+0x90/0x1b0
Jan 20 15:45:16 vm1-□ kernel: [<ffffffff8120d97c>] ? security_real_capable_noaudit+0x3c/0x70
Jan 20 15:45:16 vm1-□ kernel: [<ffffffff81113eba>] ? oom_kill_process+0x8a/0x2c0
. . .
Jan 20 15:45:16 vm1-□ kernel: [ 5963]     0  5963   361116   205625   0       0             0 mleak.pl
Jan 20 15:45:16 vm1-□ kernel: Out of memory: Kill process 5963 (mleak.pl) score 827 or sacrifice child
Jan 20 15:45:16 vm1-□ kernel: Killed process 5963, UID 0, (mleak.pl) total-vm:1444464kB, anon-
rss:822368kB, file-rss:132kB
```

  - OOM Killerによって強制停止されたプロセスが確認できます。

- カーネルパラメータ「vm.panic_on_oom」の現在の値を確認して、これを1に変更します。

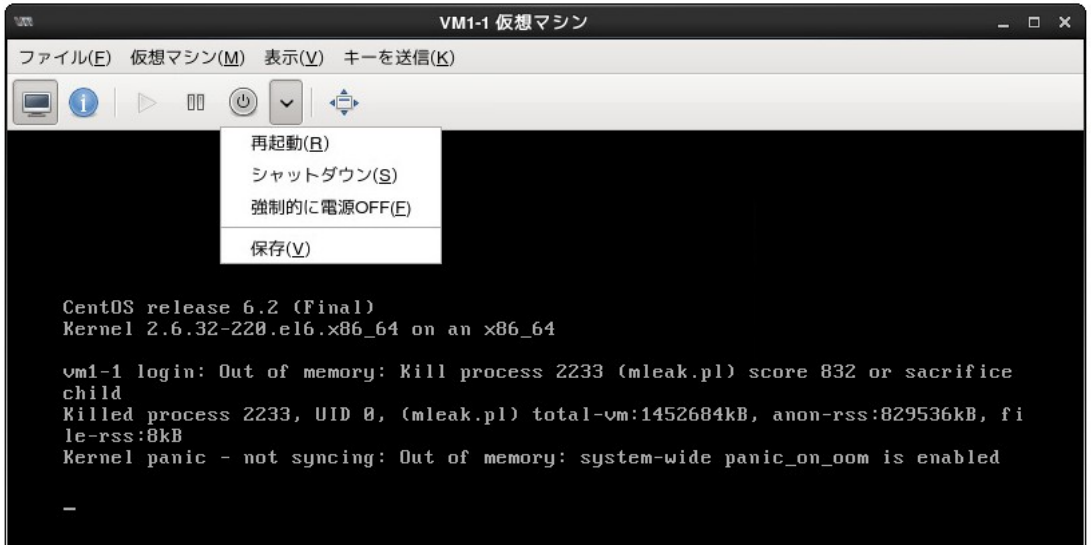
```
[root@vm1-□ ~]# cat /proc/sys/vm/panic_on_oom
0
[root@vm1-□ ~]# echo 1 > /proc/sys/vm/panic_on_oom
[root@vm1-□ ~]# cat /proc/sys/vm/panic_on_oom
1
```

OOM Killer発生状況の確認 (3)

- ホストLinuxからvirt-mangerを起動して、VM1-□の仮想コンソールを開いておきます。
- 先と同様に、スクリプトで2048MBのメモリを消費します。

```
[root@vm1-□ ~]# cd /root/work/
[root@vm1-□ work]# ./mleak.pl 2048
Consuming 2048 MB Memory...
```

  - 今回は、OOM Killerが発生したタイミングがカーネルパニックとなります。
- 仮想コンソールにカーネルパニックメッセージが表示されてることを確認します。



  - この例では、先に実行されたOOM Killerのメッセージも表示されています。
- 仮想コンソールの上図のメニューから「強制的に電源OFF」を選択して、仮想マシンを停止した後、再度、▶ボタンで仮想マシンを起動しておきます。

OOM Killer発生状況の確認 (4)

- 以上で「OOM Killer発生状況の確認」は完了です。

- 以上で「プロセス管理/メモリ管理演習」は完了です。

メモとしてお使いください

メモとしてお使いください