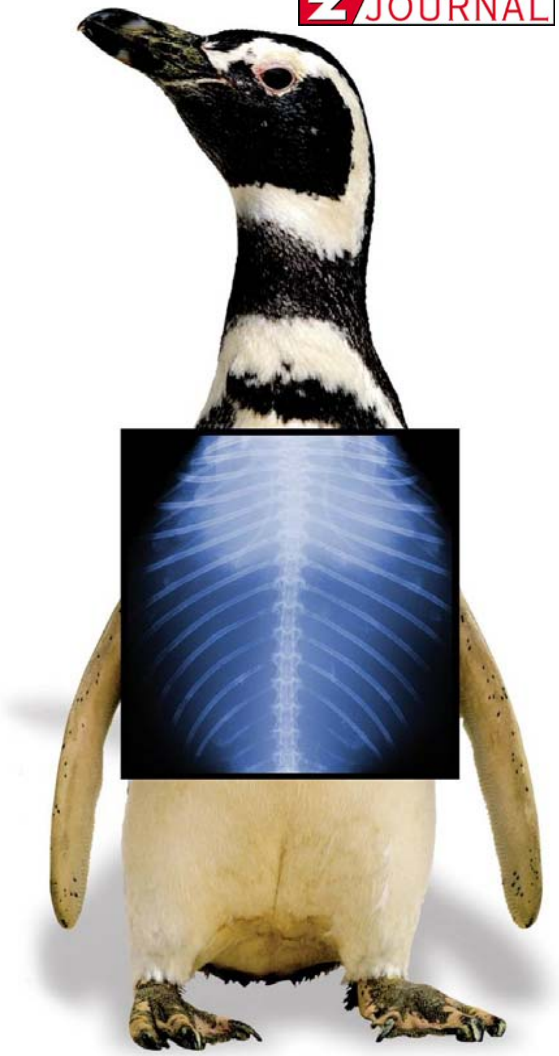
Help! My (Virtual) Penguin Is Sick!

Or

*Aptenodytes Patagonicus** Problems on z/VM

Phil Smith III

SHARE 108
February 2007
Session 9248



* King Penguin, of course!

Copyright Information

Why We're Here

The difference between
applications **people and** systems **people:**

Applications people worry about how it will work.

Systems people worry about how it will fail.

- If you support production, you're a systems person!

Agenda

- We'll cover:
  - Ways Linux can get sick
  - Techniques to decide what's wrong
  - Debugging information you can gather
- We won't cover:
  - Detailed use of debugging tools (gdb, et al.)
  - Dump (core) analysis

➢ Paramedic / First Responder functionality, not ER surgery or pathology lab forensic reports!

Penguins and Bears, Oh My!

Penguin Diseases 101

The Modal Penguin Ailment

- “Why isn’t my Linux virtual machine responding?” AKA:
  - Can I get from here to there?
  - If I can get there, is there a “there” there?
  - If there is a “there” there, is it open?

- These problems correspond to:
  - Networking problems
  - Linux issues
  - VM troubles

A Baseline is Useful!

- Linux guests vary widely
  - Networking configuration
  - Performance profile
  - Services provided
- Keep written (and online) notes about your guests
  - IP addresses, network interfaces, routing, etc.
  - Typical/observed performance characteristics
  - Disk space usage

In a crisis, you need to know how things *should* look!

Network Issues

- Is it a network issue:
  - Between the user and VM?
  - Between the VM stack and the Linux virtual machine?
  - Within the Linux virtual machine?
- If you can’t get to the machine, it sure won’t respond!

VM Troubles

- Is the Linux virtual machine even logged on?
  - Someone might have logged it off, FORCEd it, etc.
- Is the virtual machine in a stopped state?
  - Users may disconnect from machines carelessly, leaving them stopped
- Is VM broken?
  - If VM is sick, Linux sure won't run!
- Is VM letting the virtual machine run?
  - CP might not be giving it resource

Linux Issues

- Is it a kernel problem within the Linux guest?
  - Even Linux can have problems — OOMs (Out-Of-Memory errors), loops, or Oopses (kernel errors)
- Is a specific service (ssh, ftp, etc.) broken?
  - If target service is down, Linux will appear to be down
- Is it resource exhaustion within Linux?
  - Insufficient disk space, or suffering from OOMs can cause some/all Linux services to wait
  - Is an application or service hogging resources within the Linux virtual machine?

Penguin Problem Identification

Taking Your Penguin's Temperature and Pulse

Linux Diagnostic Tools

- Use Linux commands for diagnosis:
  - **`ps`** (Process Status)
  - **`df`** (Display Filesystems)
  - **`free`** (memory usage display)
  - etc…
- Many of these just display `/proc` files
  - **`/proc`** is a pseudo-filesystem whose files contain various system settings, counters, etc.
  - Better than running control blocks in memory!
  - Access files like any other file: **`cat`**, etc.
  - Write to **`/proc`** to change system settings on-the-fly

Diagnosing Network Issues

- Try to `ping` Linux from user's machine
  - Success means network OK between user & Linux
  - Helps if you know the Linux hostname/IP address
  - Also good to know whether Linux guest normally responds (some don't; some firewalls block ICMP)
- Try `traceroute` to Linux from user's machine
  - **`traceroute`** failure at last hop before Linux implicates Linux networking
  - Must know normal routing and thus normal "last hop"!
  - Linux, Windows, VM all have **`traceroute`**, spelled varying ways

Diagnosing Network Issues

- If Linux networking appears broken:
  - Log onto guest virtual machine directly
  - Then log into Linux as **`root`**
  - May not be possible if local root login disabled (may be able to login as another user and **`su`** to **`root`**)
- Use `ifconfig` and/or `netstat -i` to examine network configuration and status
  - Bouncing connection sometimes helps (**`ifconfig down`** followed by **`ifconfig up`**)

Diagnosing Network Issues (continued)

- Useful CP commands:
  - **#CP QUERY VIRTUAL NIC** shows whether virtual NICs on Guest LANs are connected
  - **#CP QUERY LAN DETAILS** shows what Guest LANs look like, including IP addresses assigned
    - Use **#CP QUERY LAN DETAILS *lanname*** if many LANs
- Try `cat /proc/net/arp`
  - Shows cached hardware addresses
  - If none, that ***may*** tell you network isn't very happy
  - Recommendation is to disable ARP caching anyway if using VSWITCH, so of limited usefulness

Diagnosing Network Issues (continued)

- If network is broadcast-capable (QDIO), `ping` the `Bcast` (broadcast) address shown by `ifconfig`:

```
ping -b -c 1 10.3.2.255
WARNING: pinging broadcast address
PING 10.3.2.255 from 10.3.2.2 : 56(84) bytes of data.
64 bytes from 10.3.2.2: icmp_seq=0 ttl=64 time=41 usec
```

  - On 3270, use **`ping –c 1`**, or **`ping`** will run forever
    - No **`<Cntrl>C`** on 3270; some distros support **`^C`**
  - More than one response from an IP address means duplicate IP!
- Learn to use `tcpdump` (or equivalent tool)
  - Beyond scope of this presentation, but very powerful!

Diagnosing VM Troubles

- Is VM broken?
  - Try to log onto another VM userid
  - If that doesn't work, head for the machine room!
- Is network to/from VM healthy?
  - Try to **`ping`** and **`traceroute`** VM from your PC
  - Try to **`ping`** external host from VM
  - If you can get out but not back in, look for routing problem external to VM
- Is the Linux virtual machine even logged on?
  - Log onto a VM userid and issue **`#CP QUERY USER`** ***`linuxid`***
  - Response ***`linuxid`*** **`NOT LOGGED ON`** is a problem!

(Digression) VM SPOOLed Consoles

- VM lets you keep a copy of all console activity for a virtual machine
  - Conceptually similar to having **`root`** logged on using a hardcopy terminal
- Files are saved in VM system SPOOL space
- Closed on demand or automatically at system shutdown or user logoff
- *Invaluable* resource for determining abnormal virtual machine events
  - A bit less useful for Linux, since most services do not log to console
  - Oopses, OOMs, some segfaults *are* logged to console

How To SPOOL the Console

- `CP SPOOL` command turns on SPOOLing:
  `CP SPOOL CONSOLE START`
- `CP TERMINAL TIMESTMP ON` useful:
  - Timestamps all output
- Various options control default destination userid, class, filename/filetype
- Useful to indicate date/time SPOOL started:
  `CP SPOOL CONSOLE START NAME` ***`yyyymmdd hh:mm:ss`***
  - Once file is closed, file timestamp will be ***close*** time, so this adds useful info
- May want to centralize console collection:
  `CP SPOOL CONSOLE START TO CONSAVER`

Finding (Open) SPOOLed Consoles

- To determine if a running virtual machine has its console SPOOLed:

  #CP QUERY PRT ALL *linuxid*

  - Look for open CON file:

```
ORIGINID FILE CLASS RECORDS  CPY HOLD DATE  TIME NAME TYPE
linuxid  6216 T CON nnnnnnn 001 NONE OPEN- 0009 name type
```

  - Mere *existence* of file is useful data point

- To close the console and send it to yourself:

  #CP SEND CP *linuxid* CLOSE CONSOLE *yourid*

  (where ***yourid*** is your userid)

  - CP **SEND** requires privilege class C

Processing VM SPOOLed Consoles

- Result of previous command is message:
  RDR FILE *nnnn* SENT FROM *linuxid* CON WAS *mmmm* RECS *rr* …
- Note the "*nnnn*" value — that's the SPOOL file number in your virtual reader
- Issue CMS PEEK command to view the file:
  PEEK *nnnn* (FOR *
  - Places you in XEDIT session, viewing file contents
  - Large files require time, virtual storage to read
  - Note: files may span days; **HCPMID6001I** appears each midnight
- CMS RECEIVE command reads file to disk
  - PF9 in **PEEK**, or:

  RECEIVE *nnnn fn ft fm*

Finding (Closed) Console Files

- To find SPOOLed consoles for non-running virtual machines (or from previous logons):

  #CP QUERY RDR ALL *linuxid*

  #CP QUERY PRT ALL *linuxid*

  - Shows files in ***linuxid*** 's virtual reader or printer

  #CP QUERY RDR ALL XFER ALL *linuxid*

  - Shows files sent/transferred to other virtual machines

- Use CP TRANSFER to move files to your reader:

  TRANSFER *ownerid* RDR *nnnn* *

  - Then use **PEEK**, **RECEIVE**, et al.

Notes About SPOOLed Consoles

- Consoles can become very large
  - For guests with significant console activity, consider closing periodically to keep files manageable
  - E.g., close at midnight via WAKEUP-based service machine
  - EOF option closes automatically every 50,000 records (desirability depends on how you manage the files)
- Naming consoles rationally helps a lot
  - Use `NAME` option when SPOOLing
  - `RECEIVE` them as "*`userid yyyymmdd`*", perhaps
- Vendor console management products exist

When/Why Was Linux Logged Off?

- Examine operator's console to see when and why it was logged off:

  `User` *`linuxid`* `LOGOFF AS` *`linuxid`* `USERS=` *`n`*

  - Logged off "normally", either by a user command or by Linux itself after shutdown

  `User` *`linuxid`* `LOGOFF AS` *`linuxid`* `USERS=` *`n`* `FORCED BY` *`vmid`*

  - Logged off by CP FORCE command issued by vmid

  `User` *`linuxid`* `LOGOFF AS` *`linuxid`* `USERS=` *`n`* `FORCED BY SYSTEM`

  - Logged off due to CP "timebomb" logoff, after being in a read for (usually) 15 minutes while disconnected

- Look for more nuggets at bottom of guest console

Diagnosing VM Troubles

- Is Linux virtual machine stopped in CP READ?
  - Issue `CP SEND CP linuxid BEGIN` to start it
    - Harmless at worst
  - Use `RUNNABLE EXEC` (see Resources) to check
- How did it get there?
  - Force disconnected with `RUN OFF`
    - by system or because user closed emulator while connected
  - Reconnected and left in `CP READ` (with `RUN OFF`)
  - `CP STOP` or `CP CPU ALL STOP` issued on guest

➢ Conclusion:

Run Linux guests with CP SET RUN ON!!!

Diagnosing VM Troubles

- Is VM giving the virtual machine any service?
  - CP might not be giving it resource
  - Likely if Linux virtual machine reconnect shows **`RUNNING`** with no keyboard response
  - If it seems normal at reconnect, hit ENTER a couple of times, look for **`VM READ`**, Linux **`login:`** prompt
  - If no read, or significant delay before login prompt, VM may not be running the virtual machine

➢ **Basic understanding of scheduling/dispatching is essential**

Scheduler and Dispatcher 101

- Virtual machines must be runnable to do work
  - CP must be willing to schedule the virtual machine
  - CP must be willing to dispatch the virtual machine
- A virtual machine is in one of three lists:
  - **Dormant** list: virtual machine has no work to do
  - **Dispatch** list: virtual machine is active and CP is allowing it to run
  - **Eligible** list: virtual machine is active, but CP is not allowing it to run
- User can also be running, of course

Scheduler and Dispatcher 101

- Scheduler decides whether there are enough resources to give a virtual machine some service
  - If not enough resources are available, virtual machine does not get scheduled
- Dispatcher gives virtual machines access to CPUs
  - If multiple virtual machines are active, they take turns
  - VM is very good at this — supports tens of thousands of active users with excellent response time

Dispatch Classes – Class 1

Class 1 virtual machines:

- Virtual machines start off as Class 1
  - Such users are usually referred to as “Q1 users”
  - CP waits one Class 1 elapsed timeslice (C1ETS) to see if it goes idle voluntarily
  - If virtual machine does not go idle within that timeslice, it is preemptively stopped from execution (“queue dropped”) — sent back to the scheduler
  - C1ETS is dynamically calculated to keep a fixed percentage of users in Class 1
  - C1ETS should be enough for short, interactive transactions (minor CMS commands)

Dispatch Classes – Class 2

Class 2 virtual machines:

- If virtual machine does not go idle voluntarily in one C1ETS, it enters Class 2
  - Such users are usually referred to as "Q2 users"
  - Next time CP runs it, it is given 8 times C1ETS
  - If virtual machine does not leave the dispatch list within that amount of time, it is queue dropped
  - Such users are presumed to be running a command, but not necessarily doing something "major"

Dispatch Classes – Class 3

Class 3 virtual machines:

- If virtual machine does not go idle voluntarily within class 2 C1ETS multiple, it enters Class 3
  - Such users are usually referred to as "Q3 users"
  - Next time CP runs it, it is given 6 times class 2 timeslice (6x8 = 48x C1ETS)
  - If virtual machine does not leave dispatch list within that amount of time, it is queue dropped
  - Such users are presumably running a long-running command

Leaving the Dispatch List

- Virtual machines leave dispatch list when they:
  - Go idle voluntarily (load a wait PSW)
  - Hold execution waiting on CP (paging, DIAGNOSE…)
  - Leave SIE emulation due to privileged instruction (privop) execution
- When virtual machine leaves dispatch list, 300ms queue drop test timer is set
  - If virtual machine resumes activity within that period, it is reinserted into previous place in queue
    - Not necessarily back to Q1!
  - Linux guests without “notimer” patch never go idle long enough to get dropped from queue!

How This Plays Out…

- CP scheduling is based on resource analysis
  - If not enough resource, virtual machines are held in Eligible list (E-list)
  - Assumption: resources will become available soon
  - If not, E-listed virtual machines ***never get scheduled***
- Dispatched virtual machines “should” go idle
  - Linux tends not to go idle (without “notimer” patch)
  - Linux virtual machines thus stay runnable all the time!
- Machines doing I/O are considered active
  - Linux machines usually have a pending network I/O
  - Fixed so network I/O now ignored for queue drop

Dispatch Classes – Class 0

- Users with `OPTION QUICKDSP` or `SET QUICKDSP ON` bypass eligible list
  - Still subject to queue drops
- Interactive users also get a Q0 stay (“hotshot”)
  - Users hitting ENTER or PF/PA keys qualify
  - Still get queue dropped, but “go to head of line” briefly
  - Return to their previous queue level after Q0 stay
- Users holding certain locks are also Q0
  - Such “lockshot” users presumably are preventing other users from running

How Does This Go Wrong?

- Linux machines tend to:
  - Be quite large (virtual storage size)
  - Have working set size close to virtual storage size
  - Stay active (rarely/never go idle)
  - Not use shared pages (DCSS)
- Linux real storage requirements are thus much higher than the average CMS virtual machine
- If enough Linux virtual machines are logged on, CP notices it will overcommit real storage
  - One or more virtual machines are E-listed — **forever!**

How Does This Manifest?

- System is running along fine
  - One guest too many is started
  - Things “just stop”
- Remember the queue drop timer:
  - Guests never go idle (as far as CP can tell)
  - Never cycle out to scheduler, so E-listed guests stay there!

Detection

- CP INDICATE QUEUES EXPANDED command:

```
LINUX902        Q3 PS  00013577/00013567 .... -232.0 A00
LINUX901        Q3 PS  00030109/00030099 .... -231.7 A00
VSCS            Q1 R   00000128/00000106 .I.. -208.7 A00
VMLINUX3        Q3 IO  00052962/00051162 .... -.9398 A00
VMLINUX3 MP01   Q3 PS  00000000/00000000 ....  .0612 A00
LINUX123        E3 R   00177823/00196608 ....  5255. A00
```

  - **HELP INDICATE QUEUES** shows meaning of output
  - CP privilege class E required
  - Virtual machine **LINUX123** is not going anywhere anytime soon…!
  - Note: “deadline time” (sixth column) sometimes very large — and very bogus

Remediation

- Buy more storage ($8K/GB — cheap!)
- Make sure “notimer” patch is enabled
  - Obviously only meaningful if guests are nominally idle
  - Remember **`cron`** et al. may wake them up anyway
- Log off some guests
- Tune guest storage sizes
  - Linux uses “extra” storage for file buffers
  - Back off virtual storage size until guests swap, then add a bit more (or not)

Diagnosing Kernel Problems

- Log onto Linux guest to see if it's even alive:
  - Hit ENTER, look for **`VM READ`**, **`login:`** prompt
  - No **`VM READ`** means Linux is "hung" (looping, E-listed, or somehow busted)
  - No login prompt could just mean **`login`** isn't running
    - Again, it helps to know what normal behavior is!
  - Look at SPOOLed console for Oops messages
- "What's an Oops?"
  - A system ABEND, in VM terms: a kernel failure
  - Like VM, may leave system in unusable state
  - Doesn't necessarily indicate code bug — faulty hardware can cause an Oops (unlikely on VM)

Basic Oops Analysis

- Utility `ksymoops` maps addresses in Oops output to kernel modules
  - Uses system map file, usually found in **`/boot`**
- Oops output used by `ksymoops` is in a file
  - Usually found in **`/var/log/messages`**
  - If **`syslogd`** not running, extract with **`dmesg`** utility (**`dmesg > oops.log`**)
  - If Linux not even that alive, cut&paste from console log, or type it back in!

➢ **If cascading Oopses, only first usually relevant**

Diagnosing Kernel Loops

- Use `#CP INDICATE USER` *`linuxid`* `EXPANDED` to watch guest CPU time
  - If increasing rapidly, guest may be looping (could just be busy, though)
  - Also note I/O counts, look for massive I/O load
- If loop suspected, log onto guest, use `CP TRACE`:

  `#CP TRACE INST RUN NOTERM PRINT`
  - Run a while; monitor with **`#CP QUERY PRT * ALL`**
  - Then issue

    `#CP TRACE END`
    `#CP CLOSE PRT *`
    …and **`RECEIVE`** the file
  - Analyze for repeated hits/patterns (or ask vendor to)

Diagnosing Broken Linux Services

- Use `ps aux` to show what services are running, pipe through grep to find target:
  - `# ps aux | grep ssh`
  - Finds any processes that mention "**ssh**" (may find the **grep** itself, too)
- Restart service that's not up and should be
  - Perhaps restart it anyway if it claims to be up but isn't responding!

Diagnosing Broken Linux Services

- Look at system log files
  - `/var/log/messages` often interesting
- `dmesg` also shows recent kernel messages
  - Looks at “kernel ring buffer”
  - Sort of like CP trace table, but just messages
- Look at logs for service in question
  - Location not predictable, alas
    - Prescribed by Linux Filesystem Hierarchy Standard, but...
    - Try `/var/log/servicename`, application directories
    - Failing that, read documentation (gasp) or code
  - Note: Linux & VM times may differ (timezone, drift)
  - Default logging levels often omit useful information
  - May need to change, wait for reoccurrence

Diagnosing Resource Exhaustion

- If Linux runs short on a resource, results “may be unpredictable”
  - Well-behaved applications will fail in graceful ways
  - Severe/rapid resource depletion may prevent this
- Nothing unique about Linux resources:
  - Disk space
  - Memory
  - Page (swap) space
  - CPU
  - All can run short!

Diagnosing Disk Space Exhaustion

- Use "df" (Display Filesystems):

```
# df -a –h
Filesystem              Size  Used Avail Use% Mounted on
none                    592M   94M  464M  17% /
none                       0     0     0   -  /proc
none                       0     0     0   -  /dev/pts
/dev/dasd/0000/part1    485M   17M  468M   4% /tmp
```

- Most interesting part is "Use%"
  - Filesystems above 90% are suspect
    - May be full due to temporary file usage
  - Again, useful to know "normal" usage levels

Diagnosing Memory Exhaustion

- Linux may take OOM errors when insufficient “real” (virtual) memory is available
  - Applications can get OOMs
  - The kernel can get an OOM too (game over!)
- OOMs are reported on Linux console:

  `Out of Memory: Killed process (`*`processname`*`)`
  (application OOM)

  `Out of memory and no killable processes`
  (kernel OOM)
- *`processname`* is same as `ps` would show
  - May or may not be actual problem process
- OOM killer configurable as of kernel level 2.4.23
  - Now applications may get individual memory allocation failures, must handle

Diagnosing Memory Exhaustion

- `free` command displays system memory use:

```
# free -t
             total       used       free shared buffers cached
Mem:        191092     185160       5932      0   13032  80548
-/+ buffers/cache:      91580      99512
Swap:       197176       2920     194256
Total:      388268     188092     200176
```

- "`-/+ buffers/cache`" line most interesting
  - Shows usage ***without*** file buffers and cache
  - Those pages reclaimable for system use (DPA, in VM terms)
  - If Swap space mostly/entirely in use, expect OOMs!

Diagnosing CPU Exhaustion

- As in most environments, a single application can grab enough CPU to slow Linux
  - Control mechanisms exist, but are not enabled by default
- `top` command is “performance monitor” tool
  - `sar` is a popular free alternative (see Resources)
  - Vendor tools exist (RMF PM, Velocity, Perfman — see Resources)
- `uptime` shows 1-, 5-, 15-minute CPU averages
  - Look for rising trend to show recent problem
  - Values above 1.00 mean CPU fully loaded (work waiting)
  - Rising values may not mean Linux is using more CPU
    - Could mean higher fraction of less available CPU

Output from top Command

```
 4:26pm  up 5 days,  7:10,  2 users,  load average: 1.00, 1.00, 1.00
82 processes: 80 sleeping, 2 running, 0 zombie, 0 stopped
CPU states:  0.8% user, 14.0% system,  0.0% nice, 85.1% idle
Mem:  191092K av, 185808K used,   5284K free,      0K shrd, 12976K buff
Swap: 197176K av,   2920K used, 194256K free                80288K cached
  PID USER  PRI  NI  SIZE  RSS SHARE STAT %CPU %MEM   TIME COMMAND
 6250 root   17   0  1060 1060   844 R     5.9  0.5   0:01 top
 6142 root    9   0  2320 2320  1828 S     0.3  1.2   0:02 sshd
    1 root    9   0   556  540   492 S     0.0  0.2   0:02 init
    2 root    9   0     0    0     0 SW    0.0  0.0   0:00 kmcheck
    3 root    9   0     0    0     0 SW    0.0  0.0   0:00 keventd
```

etc…

- Note that the top command is `top` itself!
  - Look at other candidates, note “heavy hitters”
  - “**`top d 5`**” auto-refreshes every 5 seconds, shows some trends
- See `man` page to interpret, especially STAT
  - Note “**`0.0% nice`**”
  - Negative value means some tasks have priority

Other Performance Measurements

- Look at `/proc/loadavg`
  - 4th value: #processors/#processes running (e.g, "`2/81`")
  - 5th value: # of processes started since system boot
  - Rapidly changing 5th value = something going on!
- SNMP can provide data, depending on settings
  - Must be enabled, and SNMP collector operating somewhere!
  - Do not leave default passwords (public/private strings) in place (obvious, but far too many folks do)
- Linux I/O statistics may be useful
  - Enable by `echo set on > /proc/dasd/statistics`
  - Must be enabled before problem to be useful!
  - Data saved in `/proc/dasd/statistics`

Other Performance Measurements

- `/proc/chandev` shows state of devices
  - Useful if other evidence suggests a device problem
- Learn useful CP commands:

`QUERY VIRTUAL ALL`	(lots of output!)
`QUERY VIRTUAL DASD`	(show all virtual DASD)
`QUERY VIRTUAL xxxx`	(show a specific device)
`QUERY MDISK`	(show virtual DASD ownership)

- VM performance tools provide external performance measurement
  - Can profile usage; most don't show activity inside Linux
- `iostat` (partner to `sar`) also does I/O monitoring

VM Monitor Data

- z/VM generates monitor data on demand
  - Highly granular, very efficient mechanism
- Linux for zSeries can, too
  - Data generated believed to be suspect
  - Must correlate with z/VM data to be meaningful
  - Stay tuned…

Penguin Forensics

Recording Evidence Before Burying the Body

First Failure Data Capture

- IBM promotes First Failure Data Capture:
  - Collecting useful debugging information when a problem first occurs
  - “Try a reboot” is ***not*** FFDC!
  - VM, MVS, AIX, DB2, even Tivoli push FFDC
  - Windows XP Error Reporting is (sort of) FFDC
- As Linux matures, FFDC concepts seep in
  - Logging, trace tables, memory leak/overlay traps, more dump capabilities…
  - Still mostly not standard features, however — optional installs

Log Levels

- `syslogd` (syslog daemon) collects and writes messages from various services, applications
  - Of course, it has to be running to be useful!
  - Can centralize messages from multiple systems
- Level of messages to be logged is configurable
  - Understanding logging levels for your services/applications is essential to ensuring FFDC
- Standard Linux `syslogd` isn't very smart/flexible
  - Insufficiently granular in many cases
  - Uses UDP—messages get lost due to network congestion
  - Alternatives exist, e.g., syslog-ng (**`www.balabit.com`**)

Cores

- Traditional *ix dumps were “core files”
  - Created when applications did something blatantly illegal
  - Created in current working directory, either **`core`** or **`core.pid`**
- Most distributions ship with cores disabled
  - Average wouldn’t know what to do with them!
  - May contain sensitive data from running applications
- bash `ulimit -c` size enables (current login)
  - **`ulimit -c unlimited`** means “dump everything”
  - **`ulimit -c`** displays current setting (any value > 0 = enabled)
  - See **`man bash`** for details

Dumps

- LKCD (`lcrash`) — Linux Kernel Crash Dump
  - Must be installed *before* the problem occurs
  - **`lcrash`** is the “IPCS” tool to analyze the dump
- As a VMer, I want to VMDUMP a sick penguin:
  `#CP VMDUMP 0-END TO MAINT`
  - Use IBM **`vmconvert`** to convert to LKCD format
  - VM Dump Tool is programmable, could also handle
- Standalone dump available for zSeries Linux
  - IBM mini-manual: Using the Dump Tools (LNUX-1208-01) at **`www.ibm.com/servers/eserver/zseries/os/linux/pdf/l39dmp24.pdf`**
  - Analyze standalone dumps with **`lcrash`**, too

Linux Debugging Tools

- Kernel breakpoint tools:
  - KProbes (Kernel Probes):
    `www-128.ibm.com/developerworks/library/l-kprobes.html`
  - DProbes (Dynamic KProbes):
    `sourceforge.net/projects/dprobes/`
- Kernel event (trace table) logging:
  - LTT (Linux Trace Toolkit): **`www.opersys.com/LTT/index.html`**
  - Strace (System call Trace):
    Included in most modern distros (or Google it)

More Linux Debugging Tools

- Memory debuggers:
  - YAMD (Yet Another Malloc Debugger):
    `www.cs.hmc.edu/~nate/yamd/`
  - NJAMD (Not Just Another Malloc Debugger):
    `fscked.org/proj/njamd.shtml`
- General debugger:
  - gdb (The GNU Project Debugger):
    `www.gnu.org/software/gdb/gdb.html`

Learning to Debug Linux

- Zapping Linux bugs:
  - Visit **`www.ibmsystemsmag.com`** and search
- Mastering Linux debugging techniques:
  - **`www.ibm.com/developerworks/library/l-debug/?n-l-8152`**

FFDC: What To Save

- **Linux data**
  - System log files
  - Application log files
  - Any core files
  - Application configuration files
- **VM data**
  - VM console logs
  - CP command output
  - Trace files
  - Monitor data
  - Performance monitor reports
  - Any dumps
  - Guest directory entries

Conclusion

Summary

- To the VMer, Linux is obscure and opaque
- To the Linux expert, VM is the same!
- To provide proper support, learn to use the tools
  - Both VMers and Linux folks can learn from each other
- As always, use the community
  - `linux-390@marist.edu`: zSeries Linux mailing list
  - `ibmvm@listserv.uark.edu`: z/VM mailing list

➢z/VM and Linux — even better together!

Resources

RMF PM:
`www.ibm.com/servers/eserver/zseries/zos/rmf/rmfhtmls/pmweb/pmlin.html`

Velocity Software (ESALPS): `www.velocity-software.com`

Perfman: `www.perfman.com`

sar (part of sysstat): `freshmeat.net/projects/sysstat/`

ksymoops: `www.gnu.org/directory/devel/debug/ksymoops.html`

Performance tips: `www.vm.ibm.com/perf/tips/linuxper.html`

RUNNABLE EXEC (display virtual machine status): email me

Contact Information and Credits

Contact Info

Phil Smith III

703.476.4511

phil@velocity-software.com

Thanks To…

Alex “Puffin” deVries

Scott Loveland

Neale Ferguson

Len Reed

Christopher Neufeld

Rod Stewart

Bob Thomas (z/Journal)