Help! My (Virtual) Penguin Is Sick!

Or

Aptenodytes Patagonicus*

Problems on z/VM

Phil Smith III

SHARE 109
August 2007
Session 9248

* King Penguin, of course!
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 guest 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)
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
If QDIO network, **ping** broadcast (**Bcast**) address shown by **ifconfig**:

```bash
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  nnnnnnnn  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 owneridid 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 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 yyyyymmdd”, perhaps

• Vendor console management products exist
When/Why Was Linux Logged Off?

• Examine operator’s console to see when/why guest 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

∇ Lesson:
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 and dispatching is important**
Scheduler and Dispatcher 101

• **Some critical concepts**
  - Guests must be *runnable* to do work
  - CP must be willing to schedule the guest
  - CP must be willing to dispatch the guest

• A guest is always in one of three lists:
  - **Dormant** list: guest has no work to do
  - **Dispatch** list: guest is active, CP is allowing it to run
  - **Eligible** list: guest is active, CP is not allowing it to run
  - (Can also be *running*…special case of Dispatch list!)
Scheduler and Dispatcher 101

- **CP scheduler** analyzes resources, decides whether enough to give guest service
  - Entirely storage-related (memory)
  - If not enough available, guest does not get scheduled

- **CP dispatcher** gives guests access to CPUs
  - If multiple guests 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

- When first dispatched, guest is Class 1 (“Q1”)
  - CP waits one Class 1 Elapsed Timeslice (C1ETS) to see if it goes idle voluntarily
  - Guests that do not go idle within that timeslice are preemptively stopped from execution—sent back to the scheduler
  - C1ETS is dynamically calculated to keep a fixed % of guests in class 1
  - C1ETS should be enough for short, interactive transactions (minor CMS commands)
Dispatch Classes – Class 2

• If guest does not go idle in one C1ETS, it enters Class 2 ("Q2")
  • Next time CP runs it, given 8x C1ETS
  • Guests that do not go idle within that amount of time are rescheduled
  • Such guests are presumed to be running a command, but not necessarily doing something “major”
Dispatch Classes – Class 3

• If guest does not go idle within class 2 C1ETS multiple, it enters Class 3 (“Q3”)
  • Next time CP runs it, given 6x Class 2 = 48x C1ETS
  • Guests that do not go idle within that amount of time are rescheduled
  • Such users are presumed to be running a long-running command
Dispatch Classes – Class 0

- **QUICKDSP ON** bypasses some rules
  - Still get rescheduled, but never held in eligible list

- Interactive guests (on terminals, hitting keys) also get Q0 stays (“hotshot” stays)
  - Still get rescheduled, but “go to head of line” briefly
  - Return to their previous queue level after Q0 stay
Leaving the Dispatch List

• Guests leave dispatch list because they:
  • Go idle voluntarily (load a wait PSW)
  • Wait on a CP resource (paging, DIAGNOSE I/O)
  • Leave SIE due to execution of a privileged instruction

• 300ms queue drop test timer set on dispatch list exit
  • Guest resuming activity within that period are reinserted into previous place in queue
  • Guests that don’t go idle never get queue dropped!
How This Plays Out…

- CP scheduling is based on storage analysis
  - If not enough, guests are held in **Eligible list (E-list)**
  - Assumption: other guests will go idle, storage will become available soon
  - If not, E-listed guests never get scheduled
Why This Goes Wrong

• Linux machines tend to:
  • Be quite large (virtual storage size)
  • Have working set close to virtual storage size
  • Stay active (rarely/never go idle)

• Linux real storage requirements are thus much higher than the average CMS guest

• If enough Linux guests are logged on, CP notices it will overcommit real storage
  • One or more such guests “lose”, are E-listed — and stay there!
How Does This Manifest?

- System is running along fine
  - One guest too many is started
  - Things “just stop”!
- Dispatched guests “should” go idle
  - Linux guests typically don’t, stay runnable all the time
- Historically, guests doing I/O were “active”
  - Recent releases have mostly eliminated this
- Remember the queue drop timer
  - Guests never go idle (as far as CP can tell)
  - Never get scheduled properly, so E-listing permanent!
Detection

- **CP INDICATE QUEUES EXPANDED** shows:

<table>
<thead>
<tr>
<th>Queue Name</th>
<th>Class</th>
<th>Priority</th>
<th>Resource</th>
<th>Deadline Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>LINUX902</td>
<td>Q3</td>
<td>PS</td>
<td>00013577/00013567</td>
<td>-232.0</td>
</tr>
<tr>
<td>LINUX901</td>
<td>Q3</td>
<td>PS</td>
<td>00030109/00030099</td>
<td>-231.7</td>
</tr>
<tr>
<td>VSCS</td>
<td>Q1</td>
<td>R</td>
<td>00000128/00000106</td>
<td>-208.7</td>
</tr>
<tr>
<td>VMLINUX3</td>
<td>Q3</td>
<td>IO</td>
<td>00052962/00051162</td>
<td>-.9398</td>
</tr>
<tr>
<td>VMLINUX3 MP01</td>
<td>Q3</td>
<td>PS</td>
<td>00000000/00000000</td>
<td>.0612</td>
</tr>
<tr>
<td>LINUX123</td>
<td>E3</td>
<td>R</td>
<td>00177823/00196608</td>
<td>5255.0</td>
</tr>
</tbody>
</table>

- **HELP INDICATE QUEUES** shows meaning of output
- CP privilege class E required
- **Note:** “deadline time” (sixth column) indicates when CP thinks the guest will run
- Guest **LINUX123** is not running any time soon...
Remediation

- Buy lots more storage ($<6K/GB — cheap!)
- Tune applications so guests do queue drop
  - Obviously only meaningful if guests are nominally idle
  - Remember cron et al. may wake them anyway
- Log off some guests
  - You didn’t need that WAS application, did you?
- Tune guest storage sizes
  - Linux uses “extra” storage for file buffers
  - Smaller guests may actually perform better
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)
BasicOopsAnalysis

• 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
  • 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
  - Any and 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; kernel 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 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 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` value

- Note “0.0% nice”
- Negative value would mean some tasks have priority
Other Performance Measurements

- Look at `/proc/loadavg`
  - 4th value: #processors/#processes running ("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 System z 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](http://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 user 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 z/Linux
  - Analyze standalone dumps with **lcrash**, too
Linux Debugging Tools

- Kernel breakpoint tools:
  - KProbes (Kernel Probes):
  - 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.ibm systemsmag.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: z/Linux mailing list
  • ibmvm@listserv.uark.edu: z/VM mailing list

➤ z/VM and Linux — even better together!
Resources

- 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 (virtual machine status): email me
Contact Information and Credits

Contact Info

Phil Smith III
703.568.6662
phil@velocity-software.com

Thanks To…

Alex “Puffin” devVries
Scott Loveland
Neale Ferguson
Len Reed
Christopher Neufeld
Barton Robinson
Rod Stewart
Bob Thomas (z/Journal)
Rob van der Heij