BYOL

Use z/VM to build custom Linux

Rick Troth, VSSI

<richard.troth@vsoftsys.com>

Disclaimer

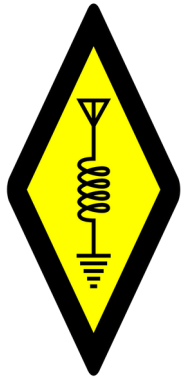
This is a discussion of personal experience.
If anything in this presentation causes you to brick your shiny new laptop, I'm truly sorry, but you may not sue me for it.

Agenda

- Introduction
- Academic Computing History
- Open Source Resources and the Standard Recipe
- Hardware Architectures
- Will try to Simplify Z Architecture for non-Z people
- Real Life Examples

about:rick

- VM since 1982
- Unix since 1985
- Pipelines since 1992
- Linux since 1993
- Knighted “Sir Santa”



Rationale for a Time Sink

- You Want Known Sources (software supply chain)
- You Need Known Behavior (security in scope)
- Good for running Offline and “Air Gapped” Systems
- Build Embedded Systems
- Own It, Control Your Own Stuff
- Fun!

Not Necessarily …

- Sole System
- Daily Driver
- Single Architecture

Okay to “borrow” packages from more complete systems.

Hobbyist Hacker History

- What's this VM thing? (yes, used VM/SP before Unix)
- Learning compiled languages … especially C
- Learning Unix (of which first was mainframe, "UTS")
- Discovered the Gnu collection and the Standard Recipe
- The Linux kernel arrives … and you can re-compile it
- Sourdough and Starter

The Unix Way

… or “The Unix Philosopy”

- Keep it Simple
- Write small programs that do one thing well
- Write programs as filters (support pipelining)
-

Linux From Scratch

- Very Thorough
- Multiple Variations
- Lots of Useful Tips/Hints/Clues
- Sometimes Too Complicated (for me)

`https://en.wikipedia.org/wiki/Linux_from_Scratch`

The Standard Recipe

For projects distributed as source code …

- Get the source (TAR, ZIP, anything) and keep it
- “Explode” the source (from ZIP or TAR, etc)
- `./configure`
- `make`
- `make install`

Chicory

A relocating build

- `make source` (*may* perform download; *will* explode fresh)
- `make config`
- `make`
- `make install` ← this is a re-directed “install”

  `/usr/opt/`*`package-version`* → *`/where/it/lives`*

  `/usr/opt/`*`package`* → *`package-version`*

CSCRATCH

An in-place build

- `make` *`package`*`.src` (*may* download; *will* explode)
- `make` *`package`*`.cfg`
- `make` *`package`*`.exe`
- `sudo make` *`package`*`.ins` ← in-place, overwrites prior

And, yes, there was once a BSCRATCH.

Should You Choose to Accept It …

- Learn Touch Typing
- Learn ‘`make`’
- Know the Standard Recipe
- Use ‘`chroot`’ (the original “container”)
- Learn (and use) different systems

Learn 'make'

- Target/Dependency Relationship
- Time Stamps Matter
- "Rules File" in current directory

Learn ‘make’

```
target:     dependency dependency dependency…
            action
            action
            action
```

Learn ‘make’

```
hello:      hello.c
            cc -o hello

hello.run:  hello
            ./hello
```

Should You Choose to Accept It …

```
$ make diffutils.exe
multiple definition of '__mktime_internal'
.../mktime.c:325: first defined here
```

- Don’t let small errors discourage you.

Some Things ya Gotta Know

- GLIBC vs Musl – “dynamic” needs a runtime

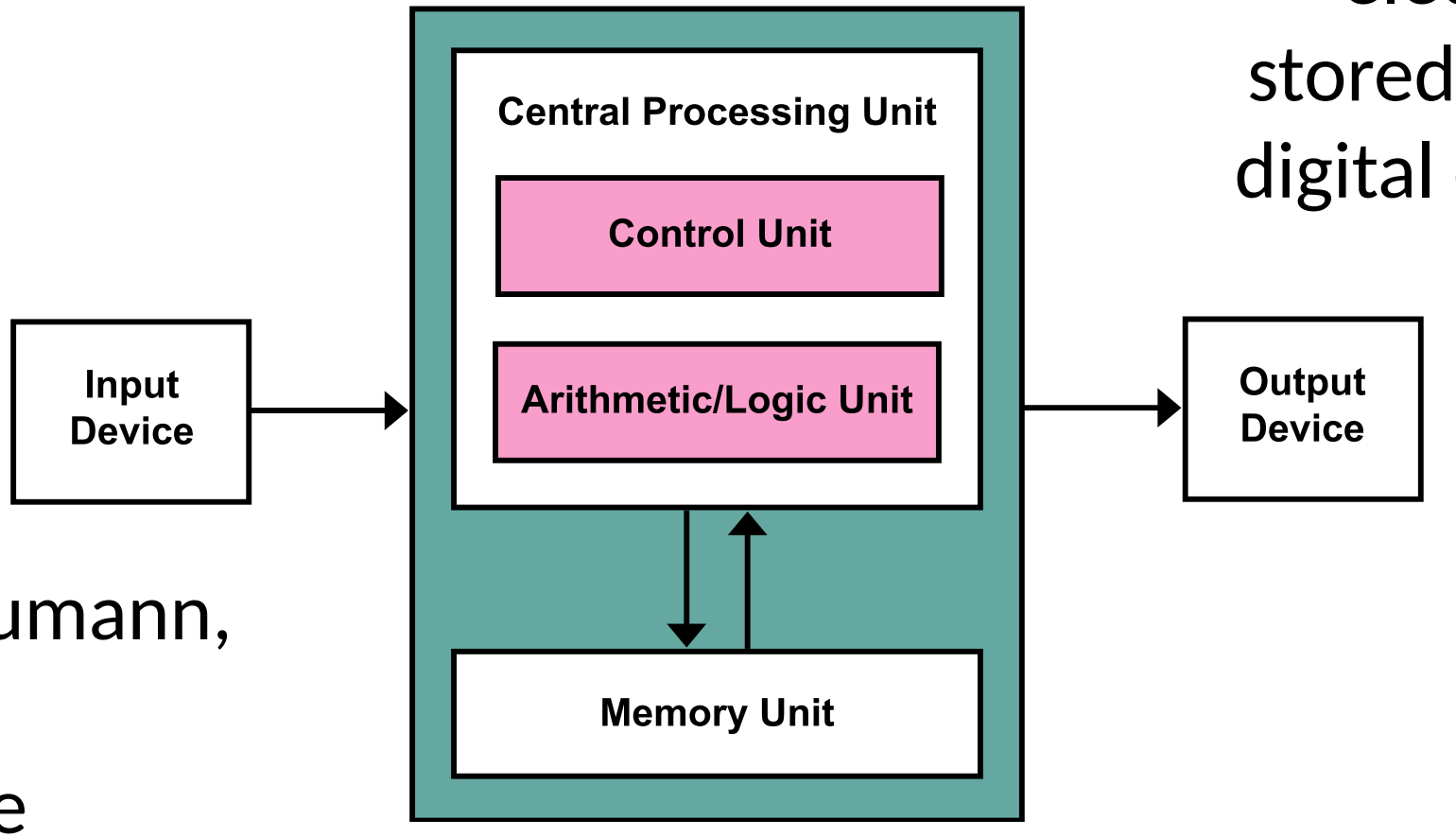
```
/usr/include
/usr/lib/*.o
/usr/lib/*.a
/lib/*ld*so*
```

Unix > Linux

- Linux began from Unix – know the universal subset
- Linux is POSIX Compliant – some standards actually work
- Linux has Advantages over other Unix*
  - more supported device types
  - partitioning is optional
  - more networking protocols
  - more filesystem types

“this is a computer”

electronic
stored-program
digital computer

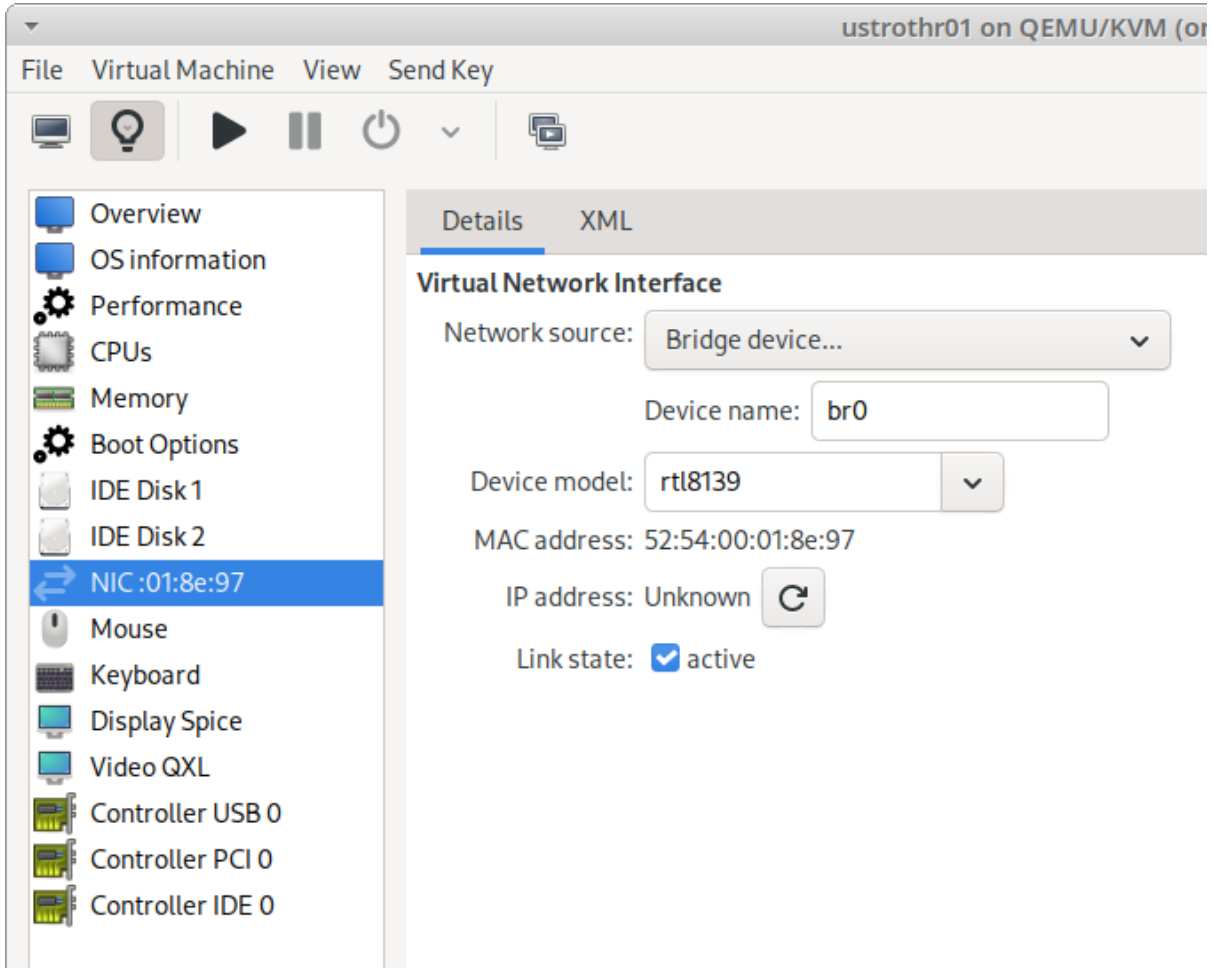


Von Neumann,
Turing,
Babbage

Z: just a computer, but better

- A CPU (one or more “cores”)
- Memory (call it “storage”)
- Disk (or equivalent) devices
- Network devices
- A console
- Channelized I/O ← what on earth is *this*?

“this is a computer”




```
STORAGE = 2G
CPU 00   ID   FF05167710900000 (BASE) CP    CPUAFF ON
CPU 01   ID   FF05167710900000 CP    CPUAFF ON
CONS 0009 3215    NOEOF OPEN                READY
RDR  000C 2540          CLOSED  EOF         READY
PUN  000D 2540    NOEOF CLOSED  OPY 001     READY
PRT  000E 1403    NOEOF CLOSED COPY 001     READY
DASD 0190 3390 VMARES R/O     214 CYL ON DASD  0104
DASD 0191 3390 VSSI02 R/W     500 CYL ON DASD  0321
DASD 019E 3390 VMARES R/O     500 CYL ON DASD  0104
DASD 01B0 3390 VSSI02 R/O    3338 CYL ON DASD  0321
DASD 01B1 3390 VSSI02 R/W     500 CYL ON DASD  0321
OSA  0600 ON NIC  0600  UNIT 000 SUBCHANNEL = 0050
     0600 DEVTYPE OSA VIRTUAL CHPID 00 OSD
     0600 MAC 02-74-00-00-00-04 CURRENT
OSA  0601 ON NIC  0600  UNIT 001 SUBCHANNEL = 0051
OSA  0602 ON NIC  0600  UNIT 002 SUBCHANNEL = 0052
```

The Console

- CONMODE 3215 ← CMS uses
- CONMODE 3270
- 3270 “back door” ← CMS uses
- Z architecture “SYSCONS”

- Z architecture “SYSASCII” (HMC only)

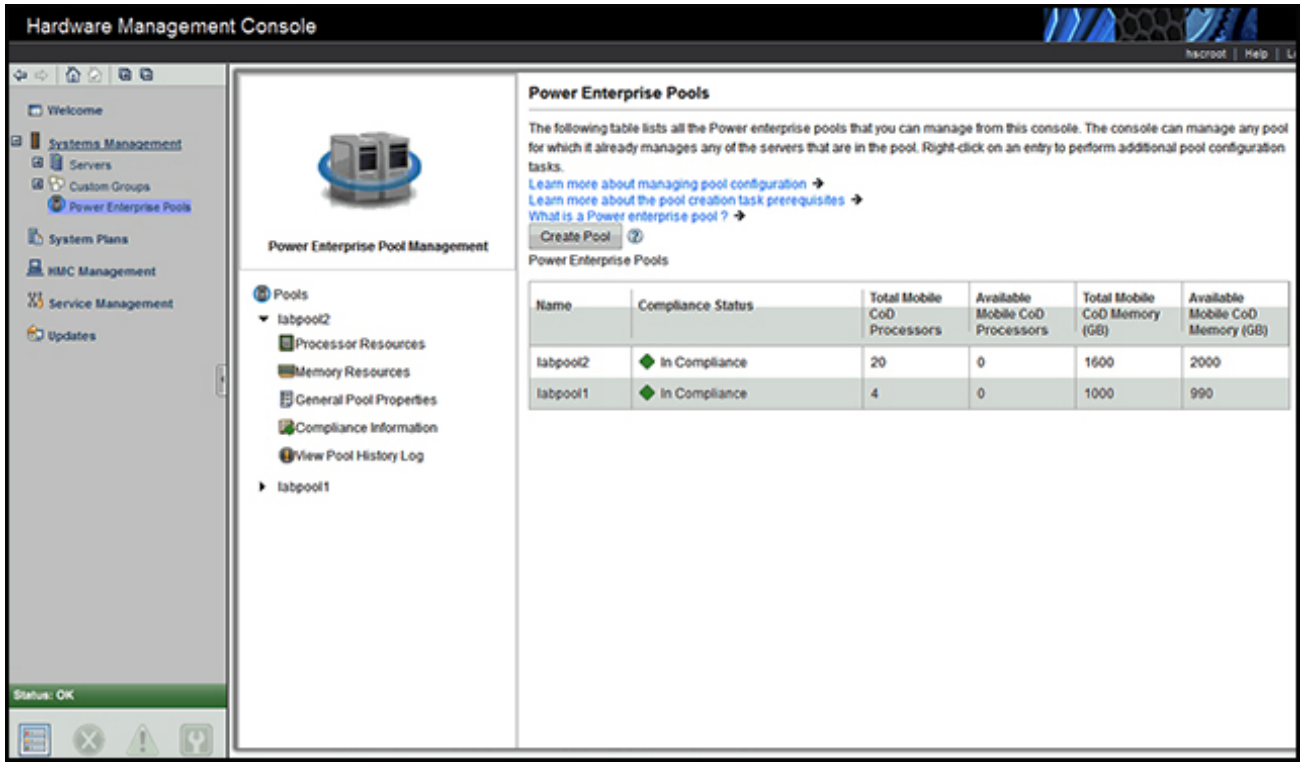
The Console

- CONMODE 3215 ← Linux uses
- CONMODE 3270 ← Linux uses
- 3270 “back door”
- Z arch “SYSCONS” ← Linux uses

- Z arch “SYSASCII” ← Linux uses?

The HMC

- Hardware Management Console
- Computer that Runs the Computer



The HMC

- Hardware Management Console
- Computer that Runs the Computer

https://en.wikipedia.org/wiki/IBM_Hardware_Management_Console

https://en.wikipedia.org/wiki/**Intel_Management_Engine**

ZNETBOOT

- Grab kernel, initrd, config, stack, IPL … one fell swoop
- Minimal CMS Involvement for newcomer, just one file
- Fast Restart if Failed Installation
- Single Configuration File, edited anywhere (Linux, Win, Mac)

Z: just a computer, but better

Linux runs on all machine types …

- Linux plus KVM on Z alongside AMD/Intel, ARM, PPC
- S390X right in the middle of it
- Don’t forget SPARC, PA-RISC, Itanium, M68K, even VAX

Z: just a computer, but better

Unix on Z …

- Linux “s390” and “s390x”
- UTS – with serious academic roots
- AIX/370 (and AIX/370 XA), AIX/ESA
- Osiris
- Solaris … until Oracle killed it

z/VM: the super smart BIOS

- CP DEFINE: STORAGE (memory), CPU, vdev AS vdev
- CP LINK owner vdev AS vdev
- CP ATTACH rdev AS vdev
- CP DETACH rdev
- CP IPL from device or from “virtual ROM”

Aggressively follows the hardware architecture.

z/VM: the super smart BIOS

- CP DISPLAY (memory contents)
- CP STORE (into memory)
- CP SPOOL CONS (or other devices) for a session record
- CP QUERY … too many things to list

All CP commands are the same whether you run CMS or Linux.

demo time

thank you!!

Rick Troth

Senior Software Developer and CISSP

<richard.troth@vsoftsys.com>

https://github.com/trothtech/nord/