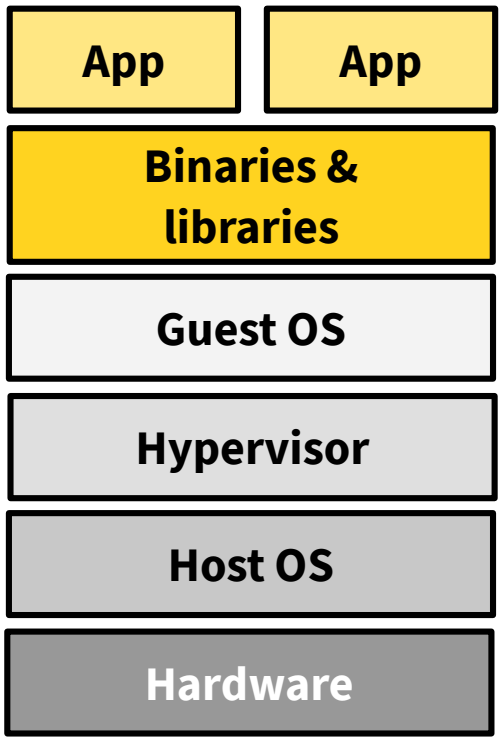
ECE 5984 Virtualization Technologies

Containers

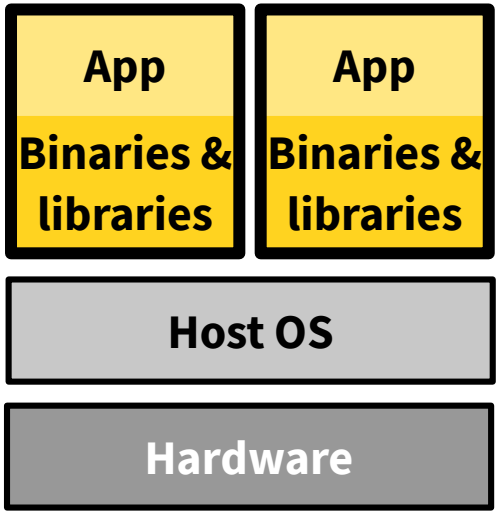
Pierre Olivier

Containers
Presentation

- **Containers: process-level sandboxing technologies**
  - Enforced by the *operating system*
    - Sometimes called *OS level virtualization*



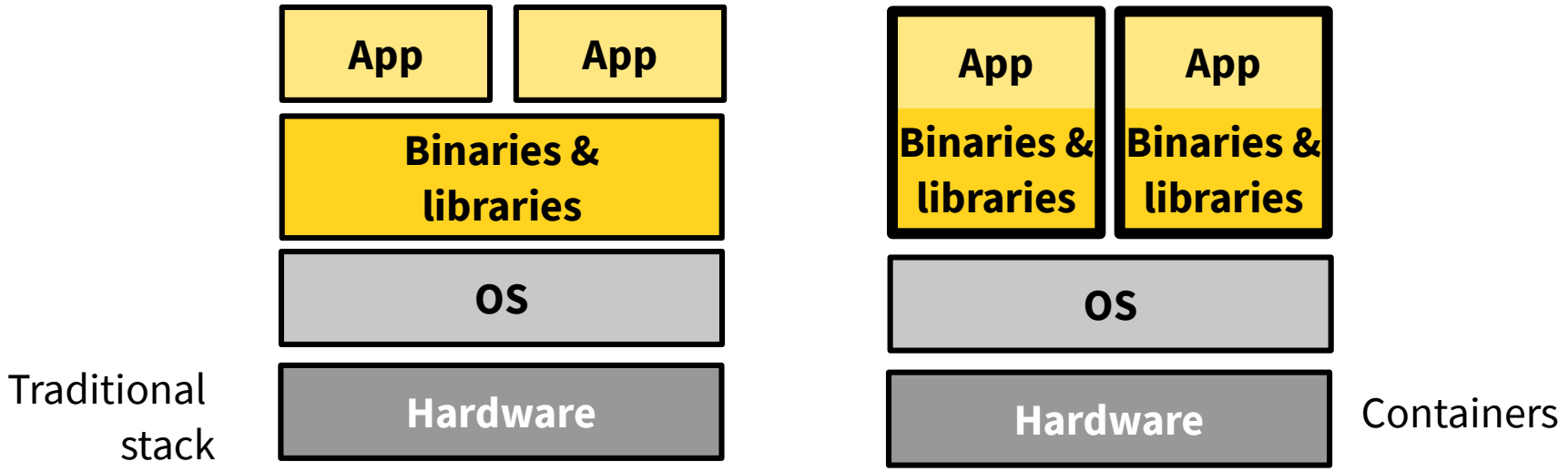
Traditional virtualization stack



Containers

Containers
Presentation

- **Containers: process-level sandboxing technologies**
  - Package application programs and dependencies
    - One of the main benefits is *ease of development/testing/deployment*
      - ➔ *"Shipping containers"*



Containers
Reason to be

Compiling HermitCore on Ubuntu 16:04 (released april 2016)
- perl version is **v5.22.1**

root@7cf989350964: ~/HermitCore/build

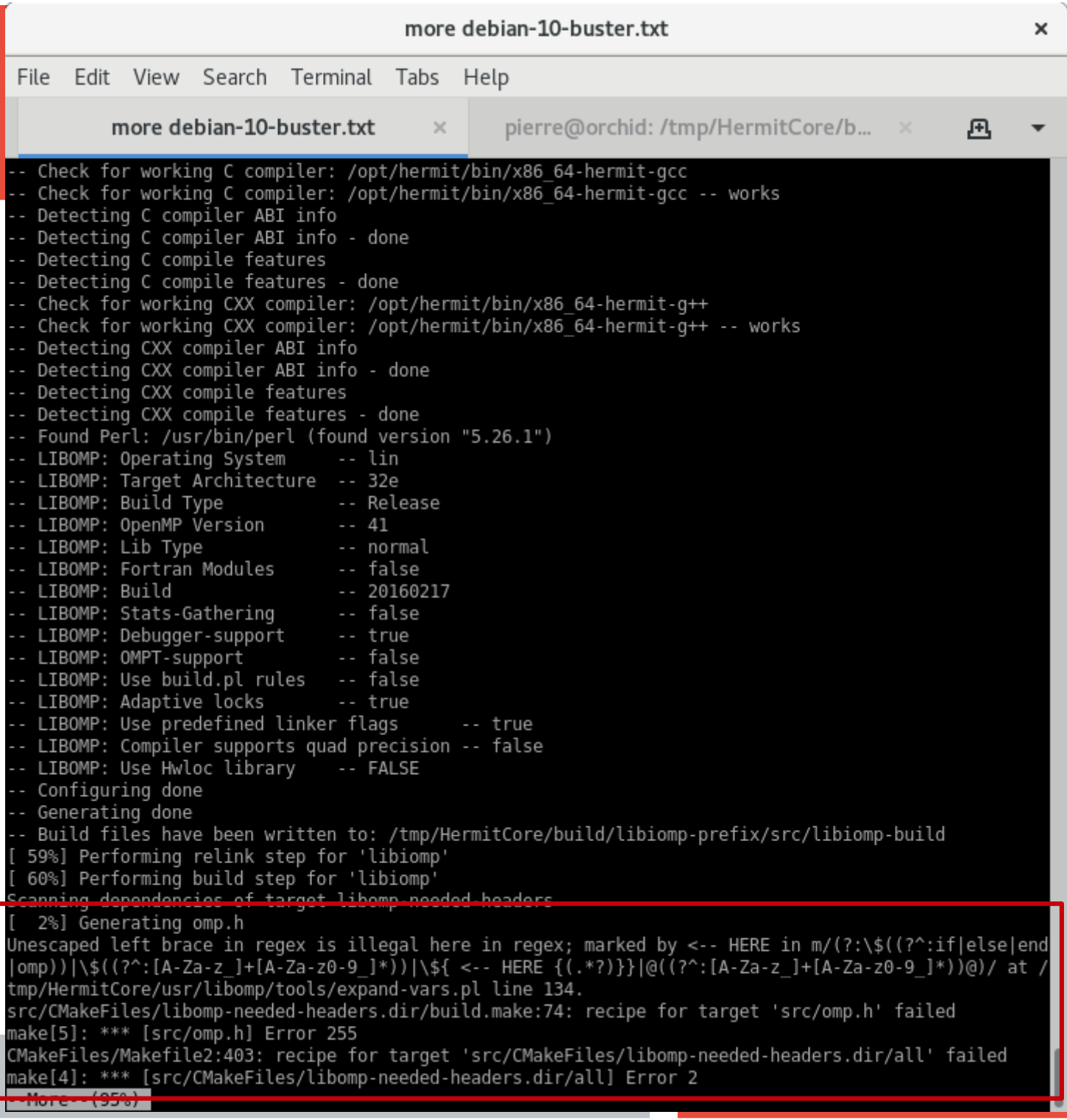
File Edit View Search Terminal Tabs Help

root@7cf989350964: ~/HermitCore/build | pierre@orchid: /tmp/HermitCore

```
-- LIBOMP: Use Hwloc library    -- FALSE
-- Configuring done
-- Generating done
-- Build files have been written to: /root/HermitCore/build/libiomp-prefix/src/libiomp-build
[ 71%] Performing relink step for 'libiomp'
[ 72%] Performing build step for 'libiomp'
Scanning dependencies of target libomp-common
[  2%] Generating omp.h
Unescaped left brace in regex is deprecated, passed through in regex; marked by <-- HERE in m/(?:\$((
?^:if|else|end|omp))|\$((?^:[A-Za-z_]+[A-Za-z0-9_]*))|\${ <-- HERE {(.*?)}}|@((?^:[A-Za-z_]+[A-Za-z0-
9_]*))@)/ at /root/HermitCore/usr/libomp/tools/expand-vars.pl line 134.
Unescaped left brace in regex is deprecated, passed through in regex; marked by <-- HERE in m/(?:\$((
?^:if|else|end|omp))|\$((?^:[A-Za-z_]+[A-Za-z0-9_]*))|\${{ <-- HERE (.*?)}}|@((?^:[A-Za-z_]+[A-Za-z0-
9_]*))@)/ at /root/HermitCore/usr/libomp/tools/expand-vars.pl line 134.
[  4%] Generating omp_lib.h
Unescaped left brace in regex is deprecated, passed through in regex; marked by <-- HERE in m/(?:\$((
?^:if|else|end|omp))|\$((?^:[A-Za-z_]+[A-Za-z0-9_]*))|\${ <-- HERE {(.*?)}}|@((?^:[A-Za-z_]+[A-Za-z0-
9_]*))@)/ at /root/HermitCore/usr/libomp/tools/expand-vars.pl line 134.
Unescaped left brace in regex is deprecated, passed through in regex; marked by <-- HERE in m/(?:\$((
?^:if|else|end|omp))|\$((?^:[A-Za-z_]+[A-Za-z0-9_]*))|\${{ <-- HERE (.*?)}}|@((?^:[A-Za-z_]+[A-Za-z0-
9_]*))@)/ at /root/HermitCore/usr/libomp/tools/expand-vars.pl line 134.
[  6%] Generating omp_lib.f
Unescaped left brace in regex is deprecated, passed through in regex; marked by <-- HERE in m/(?:\$((
?^:if|else|end|omp))|\$((?^:[A-Za-z_]+[A-Za-z0-9_]*))|\${ <-- HERE {(.*?)}}|@((?^:[A-Za-z_]+[A-Za-z0-
9_]*))@)/ at /root/HermitCore/usr/libomp/tools/expand-vars.pl line 134.
Unescaped left brace in regex is deprecated, passed through in regex; marked by <-- HERE in m/(?:\$((
?^:if|else|end|omp))|\$((?^:[A-Za-z_]+[A-Za-z0-9_]*))|\${{ <-- HERE (.*?)}}|@((?^:[A-Za-z_]+[A-Za-z0-
9_]*))@)/ at /root/HermitCore/usr/libomp/tools/expand-vars.pl line 134.
[  8%] Generating omp_lib.f90
Unescaped left brace in regex is deprecated, passed through in regex; marked by <-- HERE in m/(?:\$((
?^:if|else|end|omp))|\$((?^:[A-Za-z_]+[A-Za-z0-9_]*))|\${ <-- HERE {(.*?)}}|@((?^:[A-Za-z_]+[A-Za-z0-
9_]*))@)/ at /root/HermitCore/usr/libomp/tools/expand-vars.pl line 134.
Unescaped left brace in regex is deprecated, passed through in regex; marked by <-- HERE in m/(?:\$((
?^:if|else|end|omp))|\$((?^:[A-Za-z_]+[A-Za-z0-9_]*))|\${{ <-- HERE (.*?)}}|@((?^:[A-Za-z_]+[A-Za-z0-
9_]*))@)/ at /root/HermitCore/usr/libomp/tools/expand-vars.pl line 134.
[  8%] Built target libomp-common
Scanning dependencies of target libomp-needed-headers
[ 10%] Generating kmp_i18n_id.inc
[ 12%] Generating kmp_i18n_default.inc
[ 14%] Built target libomp-needed-headers
Scanning dependencies of target omp
[ 16%] Building C object src/CMakeFiles/omp.dir/kmp_ftn_cdecl.c.obj
                                                              208,1         31%
```

Containers
Reason to be

Compiling HermitCore
on Debian 10 Buster
(testing, 2018)
- perl version is
v5.26.1



Containers
Reason to be

- **Developing and running application X requires a complex set of dependencies**
  - Libraries sources and/or binaries (ex: glibc, etc.)
  - Build tools (ex: cmake, autotools, etc.)
  - System tools (ex: perl, grep, etc.)
  - All of these with sometimes very specific versions
- **Demo!**
  - Backup video: http://bit.ly/2F31ofC

Containers
Use cases

- **Lightweight (low cost) & elastic virtualization**
  - Containers consume few resources and can be brought up/destroyed very fast
  - Cloud services such as Gmail and Facebook make extensive use of containers
- **Development/testing**
  - Develop, build and test in a controlled, identical environment
- **Deployment**
  - Same environment as the development one (repeatability)
    - Can be deployed on any server/cloud supporting containers independently of the host configuration

Containers
Fundamental principles

- **Sandbox software running within the container**
  - *Isolate the visibility* it has on the system resources
  - Control its *resource access*

Containers
Isolated resources visibility

- **Filesystem/mount points (~chroot)**
  - Ex: can run a fedora-like rootfs on debian
  - Container cannot see host/other containers file systems
- **Network stack**
  - Container has its own IP, virtual bridged/routed network similar to VMs
- **Processes**
  - Isolated process ID set, cannot see host/other containers processes
- **IPC**
- **Hostname**
- **User IDs**
  - Can have root privileges inside container

In Linux:
Namespaces

Containers
Controlled resources access

- **Memory**
  - Limits memory and swap usage
- **CPU**
  - Limit CPU usage (can be for example 1.5 CPU) and CPU sets
  - Control CFS quotas
- **Block I/O**
  - Control throughput

In Linux:
Control groups

Containers
Different technologies

- **Chroot (1982)**
  - Generally for unix-like OS, introduced in 1982 (BSD)
  - Runtime switch to another rootfs
  - Goal: testing installation and build system of BSD
  - Chrooot isolates only the filesystem, what about isolating/controlling memory usage, network, I/O, PIDs/processes, etc.
- **FreeBSD Jails (2000)**
- **Solaris Zones (2004)**
- ***LXC*: Linux Containers (2008)**
  - High-level API controlling *Linux internal mechanisms* supporting containerization
    - Namespaces and control groups
- **Docker (2013)**
  - Another high-level API, was built on top of LXC, now using libcontainer

Container vs system-level virtual machines

- **Containers benefit:**
  - *Lightweight*
    - **Minimal resource usage** for the virtualization layer
      - ➔ All containers use the host kernel
      - ➔ Minimal disk usage (ex: Docker default ubuntu 16.04 image is ~100 MB)
    - Super-**fast startup/shutdown time → "elasticity"**
      - ➔ Starting/shutting sown a process
  - *Per-host density*
  - Nesting
- **VM benefits:**
  - Kernel versions and OS diversity
  - Performance isolation
  - Security

Containers vs unikernels

- **Lightweightness**
  - Pros & cons for both technologies
- **Security: advantage unikernels**
- **Compatibility: advantage containers**

Containers
Links

- http://www.haifux.org/lectures/320/netLec8_final.pdf
- http://www.haifux.org/lectures/299/netLec7.pdf
- https://www.cl.cam.ac.uk/~lc525/files/Linux_Containers.pdf
- https://events.static.linuxfound.org/sites/events/files/eeus13_bottomley.pdf
- http://ciecloud.csdn.net/2013/subject/07-track06-Jerome%20Petazzoni.pdf