Operating Systems

Laboratory Exercises

Semester 2, 2024-25

Course Organizer
Dr. Antonio Barbalace

Teaching Assistants
Amir Noohi
Karim Manaouil
Alan Nair

THE UNIVERSITY *of* EDINBURGH
informatics

Contents

1 Environment Setup

1.1 Introduction

Welcome to the Operating Systems laboratory exercises. This guide covers essential setup procedures for Linux kernel development, focusing on the University's virtualization infrastructure. While you may use your own setup, official support primarily covers DICE environments.

Key points:

- Local infrastructure often provides better performance than shared resources
- All setups receive equal grading consideration
- Primary support focuses on DICE environments

1.2 Using DICE Servers

The Distributed Informatics Computing Environment (DICE) provides:

- Physical lab machines in Appleton Tower
- Remote desktop access service
- Student compute servers

For DICE account setup, visit: https://computing.help.inf.ed.ac.uk/accounts

Access Methods

1. **Lab Machines:** Direct access in Appleton Tower
2. **Remote Desktop:** Instructions is available at https://computing.help.inf.ed.ac.uk/remote-desktop
3. **VPN Access:** Required for off-campus connections (https://computing.help.inf.ed.ac.uk/vpn)

1.3 Getting Your Workspace Ready

Initial Setup

After connecting to DICE:

```
# Connect to compute server
ssh student.compute

# Create workspace
mkdir /disk/scratch/sXXXXXXXX
```

Automated Setup Script

Run the provided setup script:

```
bash <(curl -sL https://os.systems-nuts.com/pre.sh)
```

Script functions:

- QEMU virtualization verification
- Directory structure creation
- Course file downloads and extraction

Example output:

```
[*] Checking for /disk/scratch...
[+] Directory /disk/scratch exists.
[*] Checking for your directory /disk/scratch/sXXXXXXX...
[*] Creating your directory /disk/scratch/sXXXXXXX...
[+] Directory /disk/scratch/sXXXXXXX created successfully.
[+] Changed to directory /disk/scratch/sXXXXXXX.
[*] Downloading files...
debian.tar.xz      100%[===================>] 344.92M  44.9MB/s
[+] Downloaded: debian.tar.xz
stop.sh            100%[===================>]   1.23K  --.-KB/s
[+] Downloaded: stop.sh
[*] Extracting debian.tar.xz...
344MiB 0:00:14 [24.4MiB/s] [============================>] 100%
[+] Extracted debian.tar.xz
[+] Removed existing debian.tar.xz
[+] Preparation Completed.
```

2 Linux Kernel Setup and Compilation

2.1 Obtaining the Kernel

Download and extract Linux kernel 6.1.74:

```
wget https://cdn.kernel.org/pub/linux/kernel/v6.x/linux-6.1.74.tar.xz
tar -xvf linux-6.1.74.tar.xz
cd linux-6.1.74
```

2.2 Kernel Configuration

Configure the kernel in three steps:

Basic Configuration

Create default x86_64 configuration:

```
make x86_64_defconfig
```

KVM Guest Optimization

Apply KVM guest settings:

```
make kvm_guest.config
```

Custom Configuration

Fine-tune settings:

```
make menuconfig
```

Required configuration options:

- `CONFIG_DEBUG_INFO` - Debug symbols
- `CONFIG_DEBUG_INFO_SPLIT` - Split debug information
- `CONFIG_GDB_SCRIPTS` - GDB helpers
- `CONFIG_KGDB` - Kernel debugging
- `CONFIG_FRAME_POINTER` - Stack traces

Recommended optimizations:

- Disable unnecessary drivers (GPU, sound, etc.)
- Disable KASLR or add `nokaslr` boot parameter

2.3 Compilation Process

Build the kernel using all available cores:

```
make -j$(nproc)
```

2.4 Compilation Outputs

Key generated files:

- **bzImage** (arch/x86/boot/):
  - Compressed kernel image
  - Primary boot file
- **vmlinux**:
  - Uncompressed kernel
  - Contains debug symbols
- **System.map**:
  - Symbol address mapping
  - Used for debugging
- **Modules** (lib/modules/):
  - Loadable kernel modules
  - Version-specific directories
- **.config**:
  - Complete configuration
  - Build reference

3 Virtual Machine Operations

3.1 Starting Your VM

Basic QEMU command structure:

```
qemu-system-x86_64 -m 4G -smp 4 \
   -drive file=debian.qcow2 \
   -kernel arch/x86/boot/bzImage \
   -append 'root=/dev/sda1 console=ttyS0' \
   -nographic
```

Parameter details:

- `-m 4G`: Memory allocation
- `-smp 4`: Virtual CPU cores
- `-drive`: Disk image specification
- `-kernel`: Kernel image path
- `-append`: Kernel parameters
- `-nographic`: Console-only mode

VM credentials:

- Username: `root`
- Password: `debian`

3.2 QEMU Monitor Usage

Access monitor with `Ctrl + a`, then `c`.

Key commands:

- `Ctrl + a h`: Help menu
- `Ctrl + a x`: Exit emulator
- `Ctrl + a s`: Save disk state
- `Ctrl + a t`: Toggle timestamps
- `Ctrl + a b`: Send break signal
- `Ctrl + a c`: Switch console/monitor

Note

For complete monitor documentation, visit:
https://qemu-project.gitlab.io/qemu/system/monitor.html

4 Kernel Debugging

4.1 Debug Preparation

Enable required kernel options:

- `CONFIG_DEBUG_INFO`
- `CONFIG_DEBUG_INFO_SPLIT`
- `CONFIG_GDB_SCRIPTS`
- `CONFIG_KGDB`
- `CONFIG_FRAME_POINTER`

4.2 Debug Setup

Launch QEMU with debugging enabled:

```
qemu-system-x86_64 -m 4G -smp 4 \
   -drive file=debian.qcow2 \
   -kernel bzImage \
   -append "root=/dev/sda1 console=ttyS0 nokaslr" \
   -nographic -S -gdb tcp::1234
```

4.3 GDB Operation

Connect and set breakpoints:

```
# Start GDB
gdb ./vmlinux

# Connect to QEMU
(gdb) target remote localhost:1234

# Set kernel entry breakpoint
(gdb) hbreak start_kernel

# Continue execution
(gdb) continue
```

5 VM Management

5.1 Runtime Control

QEMU Monitor provides essential VM control functions:

- **Basic Controls:**
  - `Ctrl + a c`: Enter/exit monitor

  - `Ctrl + a x`: Emergency shutdown
  - `Ctrl + a h`: Help menu

- **Advanced Controls:**
  - `info cpus`: Show CPU states
  - `info memory`: Memory usage
  - `info block`: Storage devices

5.2 Normal Shutdown

Preferred shutdown methods:

1. From VM console:
   ```
   # shutdown -h now
   ```
2. From QEMU monitor:
   ```
   (qemu) quit
   ```

5.3 Force Shutdown

Using `stop.sh` for unresponsive VMs:

```
[*] Detecting running QEMU Virtual Machines...
[0]  qemu-system-x86_64 -m 4G -smp 4 -drive file=debian.qcow2
-display none -display curses (PID: 546470)
Enter the index of the VM to stop: 0
[+] Sent stop signal to VM with PID 546470.
```

6 Additional Resources

6.1 Debugging Tools

Advanced debugging options:

- **printk():** Kernel message logging
- **ftrace:** Function tracing
- **kgdb:** Kernel debugger
- **Linux Oops:** Crash analysis

6.2 Documentation Links

Reference materials:

- QEMU: https://qemu-project.gitlab.io/qemu/
- Linux kernel: https://www.kernel.org/doc/html/latest/
- GDB: https://sourceware.org/gdb/documentation/

6.3 Troubleshooting

Common issues:

- **VM Freezing:**
  - Check host resources
  - Use `top` in monitor
  - Force shutdown if needed
- **Console Issues:**
  - Verify `console=ttyS0`
  - Check `-nographic` option
  - Reset terminal: `reset`

Note

Always prefer graceful shutdown methods to preserve data integrity. Use force shutdown only when necessary.