

Linux Kernel Development



Abu_Shahid

B20CS003

Abhishek_Rajora

B20CS002

Ayush_Anand

B20CS082

Objectives

01

Sandboxing
Redox OS

02

Hacking the
linux kernel

03

Adding an
additional
syscall to linux
kernel

Getting to know Redox

**Understanding
Rust and Redox**

Topic 1

**Build and
Execution**

Topic 2

Sandboxing

Topic 3

Understanding Redox



Redox is a Unix-like Operating System written in Rust.

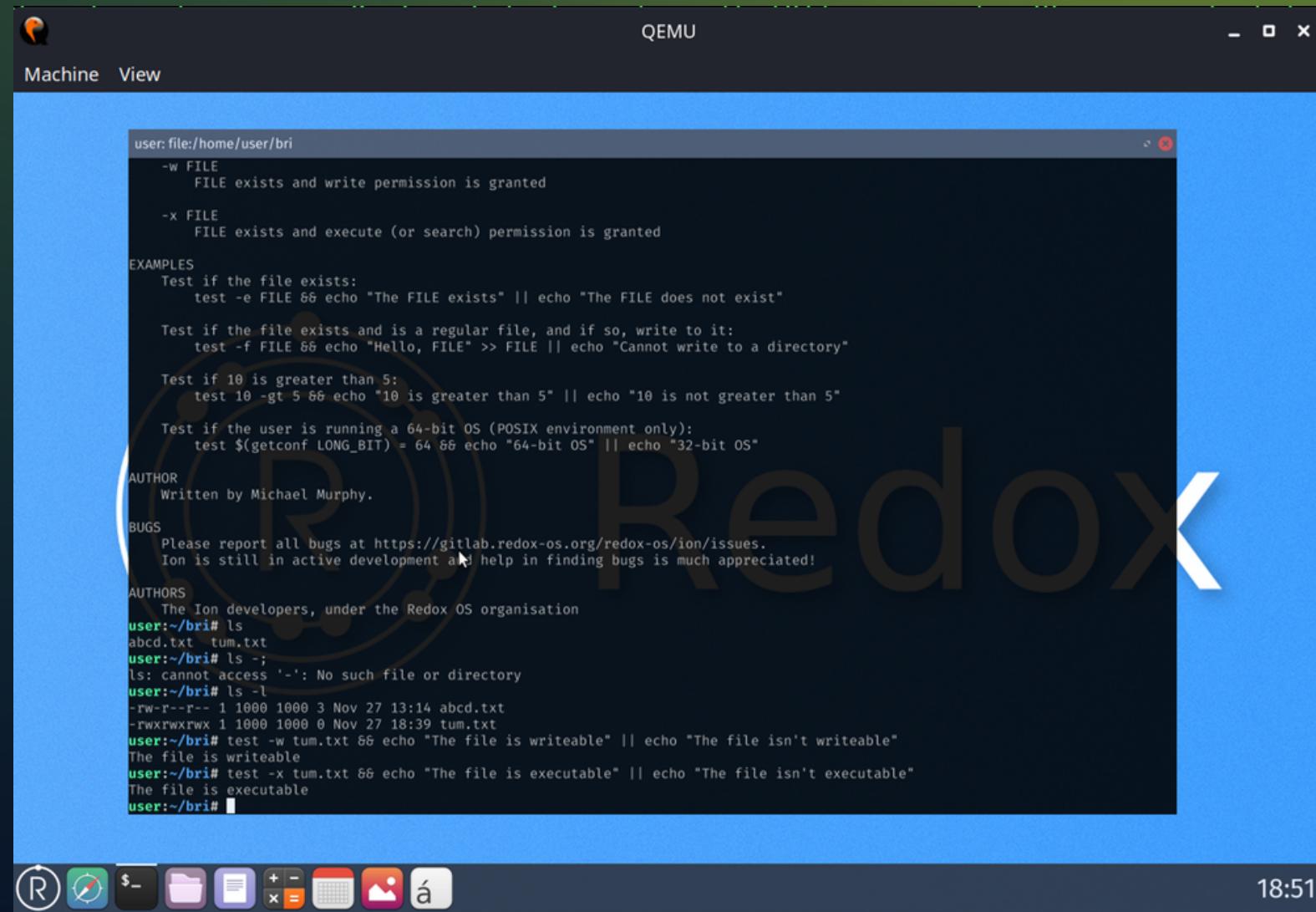


Minimal syscall interface. Only 16k lines of kernel code.



Under heavy development. Lots of features yet to be incorporated.

Rust based Implementation



The screenshot shows a terminal window titled "QEMU" running on Redox OS. The window displays a Rust-based implementation of the Ion shell. The terminal output includes documentation for file permissions (-w FILE, -x FILE), examples of file existence tests, and information about the user (Written by Michael Murphy). It also shows a bug reporting section and a transcript of a user session where they test file permissions and readability.

```
user:file:/home/user/bri
  -w FILE
    FILE exists and write permission is granted
  -x FILE
    FILE exists and execute (or search) permission is granted

EXAMPLES
Test if the file exists:
test -e FILE && echo "The FILE exists" || echo "The FILE does not exist"

Test if the file exists and is a regular file, and if so, write to it:
test -f FILE && echo "Hello, FILE" >> FILE || echo "Cannot write to a directory"

Test if 10 is greater than 5:
test 10 -gt 5 && echo "10 is greater than 5" || echo "10 is not greater than 5"

Test if the user is running a 64-bit OS (POSIX environment only):
test $(getconf LONG_BIT) = 64 && echo "64-bit OS" || echo "32-bit OS"

AUTHOR
Written by Michael Murphy.

BUGS
Please report all bugs at https://gitlab.redox-os.org/redox-os/ion/issues.
Ion is still in active development and help in finding bugs is much appreciated!

AUTHORS
The Ion developers, under the Redox OS organisation
user:~/bri# ls
abcd.txt tum.txt
user:~/bri# ls -
ls: cannot access '-': No such file or directory
user:~/bri# ls -l
-rw-r--r-- 1 1000 1000 3 Nov 27 13:14 abcd.txt
-rwxrwxrwx 1 1000 1000 0 Nov 27 18:39 tum.txt
user:~/bri# test -w tum.txt && echo "The file is writeable" || echo "The file isn't writeable"
The file is writeable
user:~/bri# test -x tum.txt && echo "The file is executable" || echo "The file isn't executable"
The file is executable
user:~/bri#
```



Follows the microkernel design and aims to be secure, usable, and free.



Rust attempts to avoid the unexpected memory unsafe conditions which are a major source of security critical bugs.



300 invocations of unsafe in about 16,000 lines of code. Carefully audited to ensure correctness.

Redox OS:

How to setup

Redox?

```
2022-11-27T23-44-58..202++00:00  [@ihdad::hda::device:379 INFO] Addr: 00:04, Type: AudioInput, Inputs: 1: [(0, 5)].  
2022-11-27T23-44-58..233++00:00  [@ihdad::hda::device:379 INFO] Addr: 00:05, Type: PinComplex: LineIn, Inputs: 0: [].  
2022-11-27T23-44-58..262++00:00  [@ihdad::hda::device:471 INFO] Best pin: 0:03  
2022-11-27T23-44-58..282++00:00  [@ihdad::hda::device:478 INFO] Path to DAC: [(0, 3), (0, 2)]  
2022-11-27T23-44-58..315++00:00  [@ihdad::hda::device:490 INFO] Supported Formats: 000201FC  
2022-11-27T23-44-58..339++00:00  [@ihdad::hda::device:491 INFO] Capabilities: 00400101  
2022-11-27T23-44-58..362++00:00  [@ihdad::hda::device:212 INFO] IHDA: Initialization finished.  
2022-11-27T23-44-58..390++00:00  [@pcid:278 INFO] PCI 00/03/00 8086:100E 02.00.00.03 Network 0=FEBC0000 1=C000  
2022-11-27T23-44-58..410++00:00  [@pcid:401 INFO] PCI DEVICE CAPABILITIES for e1000d: []  
2022-11-27T23-44-58..439++00:00  [@pcid:463 INFO] PCID SPAWN "e1000d" "pci-00.03.00" "FEBC0000" "00020000" "11"  
+ E1000 pci-00.03.00_e1000 on: FEBC0000 size: 131072 IRQ: 11  
- MAC: 52:54:00:12:34:56  
- Link is up with speed 1000 Mb/s  
2022-11-27T23-44-58..623++00:00  [@pcid:278 INFO] PCI 00/04/00 1033:0194 0C.03.30.03 SerialBus XHCI 0=00000000FEBF4000  
2022-11-27T23-44-58..848++00:00  [@pcid:278 INFO] PCI 00/1F/00 8086:2918 06.01.00.02 Bridge  
2022-11-27T23-44-58..864++00:00  [@pcid:278 INFO] PCI 00/1F/02 8086:2922 01.06.01.02 Storage SATA AHCI 4=C080 5=FEBF90  
00  
user:~# ls  
bri work  
user:~# exit  
##### Redox OS #####  
# Login with the following: #  
# `user` #  
# `root`:`password` #  
#####  
  
redox login: user  
Welcome to Redox OS!  
  
user:~#
```

How to setup Redox OS for kernel development?

install rust toolchain

```
$ curl https://sh.rustup.rs -  
  sSf | sh  
$ source  
HOME/.cargo/env
```

Clone the source

```
$ git clone  
https://gitlab.redox-os.org/redox-os/redox.git --origin  
upstream --recursive
```

Compile and build the kernel

```
$ cargo install xargo  
cargo-config  
$ git submodule update --  
recursive --init  
$ make all  
$ make qemu
```

Linux Kernel Development

Adding a syscall to the linux-
6.0 kernel

The epicenter of open source development

Linux Kernel 6.0

Starting from linux-6 parts are being rewritten in rust

One of the most active open source projects

Setting up Linux kernel

clone the source

```
git clone  
git://git.kernel.org/p  
ub/scm/linux/kernel/gi  
t/torvalds/linux.git
```

Compile and build the kernel image

```
$make defconfig  
$make oldconfig  
$make -j8
```

Mount a filesystem

**TO EMULATE THE KERNEL IMAGE IN
QEMU EMULATOR WE NEED A
FILESYSTEM, TO DO THAT WE USE
SYZKALLER SCRIPT**

```
./create-image.sh
```

SYSCALL

Adding a syscall to the linux-6.0 kernel

```
root@syzkaller:~# ./a.out
[ 108.762614] Hi I am Ayush here :)
0
root@syzkaller:~# dmesg | tail
[ 11.260781] systemd[1]: Started Load/Save Random Seed.
[ 12.082287] systemd[1]: Started Journal Service.
[ 13.246477] audit: type=1107 audit(1669591791.263:5): pid=1 uid=0 auid=4294967295 ses=t:s0 msg='avc: denied { stop } for auid=n/a uid=0 gid=0 path="/lib/systemd/system/systemctl --flush" scontext=system_u:system_r:kernel_t:s0 tcontext=system_u:object_r:unlabeled exe="/lib/systemd/systemd" sauid=0 hostname=? addr=? terminal=?'
[ 13.303204] systemd-journald[103]: Received request to flush runtime journal from PID
[ 20.471282] audit: type=1107 audit(1669591798.490:6): pid=1 uid=0 auid=4294967295 ses=t:s0 msg='avc: denied { status } for auid=n/a uid=0 gid=0 path="/lib/systemd/system/grand-update-utmp runlevel" scontext=system_u:system_r:kernel_t:s0 tcontext=system_u:object_r:unlabeled exe="/lib/systemd/systemd" sauid=0 hostname=? addr=? terminal=?'
[ 23.761205] e1000: eth0 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: RX
[ 23.824616] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
[ 108.762614] Hi I am Ayush here :)
root@syzkaller:~#
```

```
#include<stdio.h>
#include<unistd.h>
#include<sys/syscall.h>

int main() {
    long res = syscall(451);
    printf("%d\n",res);
    return res;
}
```

Creating a syscall overview

**Update the syscall
table with a new
entry**

```
$cd arch/x86/entry/syscalls/  
$vi syscall_64.tbl  
"451 common hello_there  
sys_hello_there"
```

**Implement the
function in
kernel/sys.c**

```
SYSCALL_DEFINE0(hello_there){  
    printk(KERN_INFO "xyz");  
    return 0;  
};
```

**Make the source
again**

**COMPILE AND BUILD THE KERNEL
IMAGE AGAIN**

Thank you