




Drivers From Outer Space

Fast, Simple Driver Development

Daniel Maslowski

Agenda






-  Hardware and Driver Issues
-  From Outer Space...?
-  Seamless Revolution

Introduction






Hello, I am Daniel :-)



Work and education

-  IT security and computer science
-  software engineer
-  web and mobile apps
-  infrastructure, UIs
-  ecommerce, emergency calls

Open Source contributions

-  hardware and firmware
-  operating systems
-  software distributions
-  reverse engineering
-  Fiedka the Firmware Editor

Hardware and Driver Issues

Products in the Wild

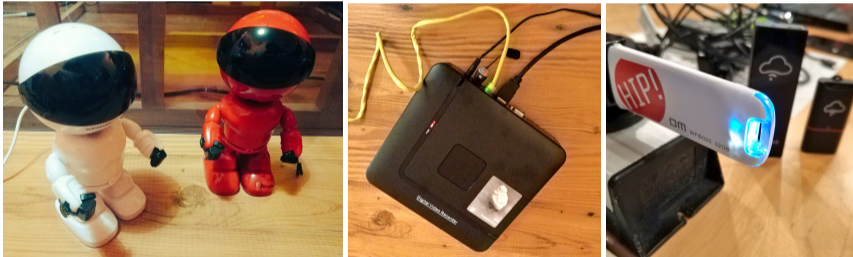
Products in the Wild

There are many hardware products that never receive any updates.

Products in the Wild

There are many hardware products that never receive any updates.

Take these gadgets for example:



Related talk:

<https://metaspora.org/repurposing-gadgets-fossasia2021.pdf>

Embedded Devices

Embedded Devices

They are usually built around microcontrollers and/or SoCs.

Embedded Devices

They are usually built around microcontrollers and/or SoCs.

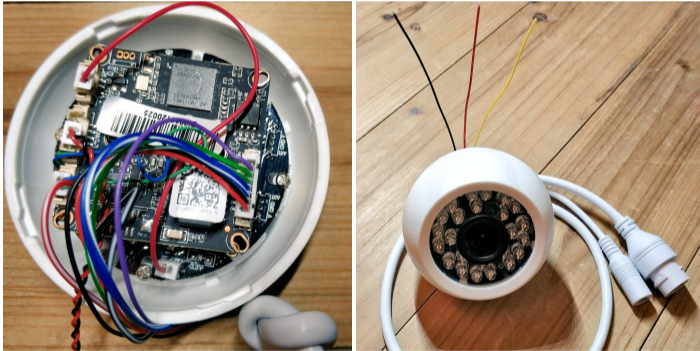
They often run Linux! :-)

Embedded Devices

They are usually built around microcontrollers and/or SoCs.

They often run Linux! :-)

Let's look inside...



... and solder some wires.

Linux in Firmware

Linux in Firmware

Not a novel idea.

Linux in Firmware

Not a novel idea.

We're doing this in many projects!



Linux in Firmware

Not a novel idea.

We're doing this in many projects!



coreboot



LinuxBoot



openembedded

coreboot was originally LinuxBIOS :-)

Where are the drivers?

Where are the drivers?

Lots of drivers are in mainline Linux, as in, kernel.org.

Where are the drivers?

Lots of drivers are in mainline Linux, as in, kernel.org.

However, many vendors do not upstream, nor publish their drivers.

Where are the drivers?

Lots of drivers are in mainline Linux, as in, kernel.org.

However, many vendors do not upstream, nor publish their drivers.

If you are lucky, you may find an SDK (software development kit) on the web.

Where are the drivers?

Lots of drivers are in mainline Linux, as in, kernel.org.

However, many vendors do not upstream, nor publish their drivers.

If you are lucky, you may find an SDK (software development kit) on the web.

Commonly, their quality is rather low and “just okay”:



prebuilt legacy 32bit toolchains



kernel source dumps as zip files instead of git repos



drivers causing compiler warnings, hacked into existing code



i.e., not upstreamable without reworking them




Where are the drivers?

Lots of drivers are in mainline Linux, as in, kernel.org.

However, many vendors do not upstream, nor publish their drivers.

If you are lucky, you may find an SDK (software development kit) on the web.

Commonly, their quality is rather low and “just okay”:

-  prebuilt legacy 32bit toolchains
-  kernel source dumps as zip files instead of git repos
-  drivers causing compiler warnings, hacked into existing code
 - ▶ i.e., not upstreamable without reworking them

Some are not open source. Linux logs proprietary drivers as “tainting”.

Where are the drivers?

Lots of drivers are in mainline Linux, as in, kernel.org.

However, many vendors do not upstream, nor publish their drivers.

If you are lucky, you may find an SDK (software development kit) on the web.

Commonly, their quality is rather low and “just okay”:



prebuilt legacy 32bit toolchains



kernel source dumps as zip files instead of git repos



drivers causing compiler warnings, hacked into existing code



i.e., not upstreamable without reworking them

Some are not open source. Linux logs proprietary drivers as “tainting”.

As a customer making products, you are often required to sign NDAs in order to receive the SDK from the vendor.

Where are the drivers?

Lots of drivers are in mainline Linux, as in, kernel.org.

However, many vendors do not upstream, nor publish their drivers.

If you are lucky, you may find an SDK (software development kit) on the web.

Commonly, their quality is rather low and “just okay”:



prebuilt legacy 32bit toolchains



kernel source dumps as zip files instead of git repos



drivers causing compiler warnings, hacked into existing code



i.e., not upstreamable without reworking them

Some are not open source. Linux logs proprietary drivers as “tainting”.

As a customer making products, you are often required to sign NDAs in order to receive the SDK from the vendor.

So developers cannot even work with upstream/mainline Linux.

From Outer Space...?

Bell Labs, the 90ies

Bell Labs, the 90ies

Plan 9 from Bell Labs, a research operating system



Bell Labs, the 90ies

Plan 9 from Bell Labs, a research operating system



Yes, the name is a reference to the Ed Wood 1959 cult science fiction Z-movie *Plan 9 from Outer Space*.

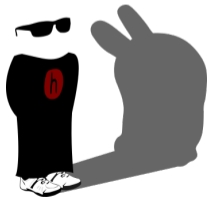
Bell Labs, the 90ies

Plan 9 from Bell Labs, a research operating system



Yes, the name is a reference to the Ed Wood 1959 cult science fiction Z-movie *Plan 9 from Outer Space*.

I haven't seen it. But I've run into the OS. :-)



Plan 9 Concepts

Plan 9 is a *network based* operating system.

Plan 9 Concepts

Plan 9 is a *network based* operating system.

Four primitives

Plan 9 Concepts

Plan 9 is a *network based* operating system.

Four primitives



file server: serving files, like NFS, though based on 9P

Plan 9 Concepts

Plan 9 is a *network based* operating system.

Four primitives



file server: serving files, like NFS, though based on 9P



authentication server: can be compared with modern IAM

Plan 9 Concepts

Plan 9 is a *network based* operating system.

Four primitives



file server: serving files, like NFS, though based on 9P



authentication server: can be compared with modern IAM



cpu server: defining a machine as *arbitrary compute resource*

Plan 9 Concepts

Plan 9 is a *network based* operating system.

Four primitives



file server: serving files, like NFS, though based on 9P



authentication server: can be compared with modern IAM



cpu server: defining a machine as *arbitrary compute resource*



terminal: what connects to a cpu server, sends commands

Plan 9 Concepts

Plan 9 is a *network based* operating system.

Four primitives



file server: serving files, like NFS, though based on 9P



authentication server: can be compared with modern IAM



cpu server: defining a machine as *arbitrary compute resource*



terminal: what connects to a cpu server, sends commands

What do we make out of this?

Plan 9 Concepts

Plan 9 is a *network based* operating system.

Four primitives



file server: serving files, like NFS, though based on 9P



authentication server: can be compared with modern IAM



cpu server: defining a machine as *arbitrary compute resource*



terminal: what connects to a cpu server, sends commands

What do we make out of this?

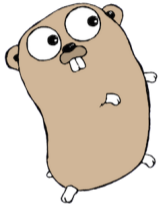
We are going to apply the idea behind `cpu` to Linux.

The new cpu

Uses SSH for authentication and command transport, 9p or optionally other means for file transport.

The new cpu

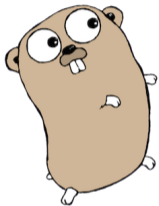
Uses SSH for authentication and command transport, 9p or optionally other means for file transport.



Written in Go, easily portable

The new cpu

Uses SSH for authentication and command transport, 9p or optionally other means for file transport.



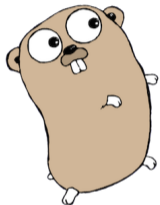
Written in Go, easily portable



Second implementation in Rust WIP

The new cpu

Uses SSH for authentication and command transport, 9p or optionally other means for file transport.



Written in Go, easily portable

Sources and Development

<https://github.com/u-root/cpu>

<https://book.linuxboot.org/cpu>



Second implementation in Rust WIP

Seamless Revolution

Run a command over `cpu`

Run a command over `cpu`

Remote target: IP camera running a Linux kernel and `cpu`. See also:

<https://github.com/orangecms/arm-cpu>



`hisilicon/HI3516EV200` contains `Makefile` etc

Run a command over cpu

Remote target: IP camera running a Linux kernel and cpud. See also:

<https://github.com/orangecms/arm-cpu>



hisilicon/HI3516EV200 contains Makefile etc

Grab ipctool from the OpenIPC project to investigate a bit.

<https://github.com/openipc/ipctool>



<https://openipc.org>

DEMO: Run commands over `cpu`

What did we just do?

What did we just do?

```
CPU_NAMESPACE=/home ../cpu -key ~/.ssh/cpu_rsa 192.168.0.222 \  
../bin/ipctool -c
```


What did we just do?

```
CPU_NAMESPACE=/home ../cpu -key ~/.ssh/cpu_rsa 192.168.0.222 \  
../bin/ipctool -c
```



CPU_NAMESPACE=/home: we have the remote mount *our* /home
onto theirs

What did we just do?

```
CPU_NAMESPACE=/home ../cpu -key ~/.ssh/cpu_rsa 192.168.0.222 \  
../bin/ipctool -c
```






CPU_NAMESPACE=/home: we have the remote mount *our* /home onto theirs



../cpu: we run the `cpu` command from our machine, as we do with many others





What did we just do?

```
CPU_NAMESPACE=/home ../cpu -key ~/.ssh/cpu_rsa 192.168.0.222 \  
../bin/ipctool -c
```

-  CPU_NAMESPACE=/home: we have the remote mount *our* /home onto theirs
-  ../cpu: we run the cpu command from our machine, as we do with many others
-  -key ~/.ssh/cpu: we pass our SSH key, can be implicit by convention






What did we just do?

```
CPU_NAMESPACE=/home ../cpu -key ~/.ssh/cpu_rsa 192.168.0.222 \  
  ../bin/ipctool -c
```

-  CPU_NAMESPACE=/home: we have the remote mount *our* /home onto theirs
-  ../cpu: we run the `cpu` command from our machine, as we do with many others
-  -key ~/.ssh/cpu: we pass our SSH key, can be implicit by convention
-  192.168.0.222: the address of the remote machine to run on, running `cpud`

What did we just do?

```
CPU_NAMESPACE=/home ../cpu -key ~/.ssh/cpu_rsa 192.168.0.222 \  
../bin/ipctool -c
```

-  CPU_NAMESPACE=/home: we have the remote mount *our* /home onto theirs
-  ../cpu: we run the `cpu` command from our machine, as we do with many others
-  -key ~/.ssh/cpu: we pass our SSH key, can be implicit by convention
-  192.168.0.222: the address of the remote machine to run on, running `cpud`
-  ../bin/ipctool -c: the command to run on the remote, coming *from us*

DEMO: Relationships between host and remote

Load a Driver via cpu

Note: This command is abbreviated to focus on the essential point.

```
./cpu camera /sbin/insmod ./lib/modules/sys_config.ko
```

Load a Driver via cpu

Note: This command is abbreviated to focus on the essential point.

```
./cpu camera /bbin/insmod ./lib/modules/sys_config.ko
```

This means that we can write apps *and* drivers *locally*, and run them on the remote *right away* without explicit copying, NFS shares, USB sticks, etc! :-)

DEMO: Interfacing with remote devices

Playing Along in a VM

Playing Along in a VM

```
git clone https://github.com/u-root/cpubinaries  
cd cpubinaries  
./QEMU -kernel cpukernel
```

Playing Along in a VM

```
git clone https://github.com/u-root/cpubinaries
cd cpubinaries
./QEMU -kernel cpukernel
```

In another session:

```
cd cpubinaries
./cpu -key ./cpu_rsa localhost cat /proc/cpuinfo
processor          : 0
vendor_id         : AuthenticAMD
cpu family       : 6
model            : 6
model name       : QEMU TCG CPU version 2.5+
stepping        : 3
microcode       : 0x1000065
cpu MHz         : 2415.355
```

Related Ideas

Related Ideas

Hellaphone

Researchers put Inferno (another Plan 9 OS) instead of Android's Java environment on a phone. Inferno can run as a Linux userland.

Related Ideas

Hellaphone

Researchers put Inferno (another Plan 9 OS) instead of Android's Java environment on a phone. Inferno can run as a Linux userland.

In Plan 9, there is a strong "everything is a file" (for real) concept. I.e., you can echo `"dial +123456789" > /phone/phone`.

Related Ideas

Hellaphone

Researchers put Inferno (another Plan 9 OS) instead of Android's Java environment on a phone. Inferno can run as a Linux userland.

In Plan 9, there is a strong "everything is a file" (for real) concept. I.e., you can echo "dial +123456789" > /phone/phone.

So from another machine, you can also do this:

```
echo "sms +123456789 Hello!" | cpu phone tee /phone/sms
```


Related Ideas

Hellaphone

Researchers put Inferno (another Plan 9 OS) instead of Android's Java environment on a phone. Inferno can run as a Linux userland.

In Plan 9, there is a strong "everything is a file" (for real) concept. I.e., you can echo "dial +123456789" > /phone/phone.

So from another machine, you can also do this:

```
echo "sms +123456789 Hello!" | cpu phone tee /phone/sms
```

Put that on a PinePhone, make it a 9inePhone! :-)

Related Ideas

Hellaphone

Researchers put Inferno (another Plan 9 OS) instead of Android's Java environment on a phone. Inferno can run as a Linux userland.

In Plan 9, there is a strong "everything is a file" (for real) concept. I.e., you can echo "dial +123456789" > /phone/phone.

So from another machine, you can also do this:

```
echo "sms +123456789 Hello!" | cpu phone tee /phone/sms
```

Put that on a PinePhone, make it a NinePhone! :-)

On a Raspberry Pi (remember: not very open!)



article: Poor Man's Virtual FileSystem with 9p, Rust, and a Raspberry Pi



pick up Gokrazy and add cpud



just run cpud in a stock Raspberry Pi OS environment

Thanks!

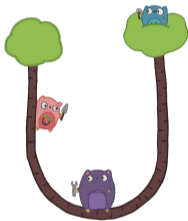
Questions?

Resources

Slides

<https://metaspora.org/drivers-from-outer-space.pdf>

Projects



<https://u-root.org>



LinuxBoot

<https://linuxboot.org>

Repos, LinuxBoot chapter

<https://github.com/u-root/cpu>

<https://github.com/u-root/cpubinaries>

<https://book.linuxboot.org/cpu>