



RUNNING LINUX ON YOUR SMARTPHONE

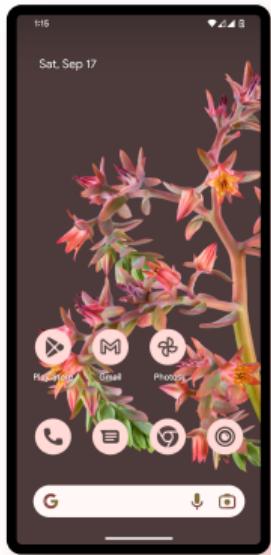
REG Lunchtime Tech Talk

David Llewellyn-Jones
9th May 2023



SAILFISH OS

SMARTPHONE OPERATING SYSTEMS

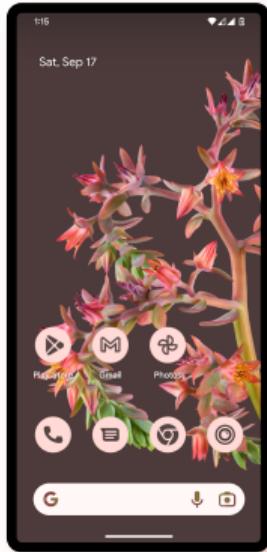


android
A stylized green Android robot head with two white eyes and a smiling mouth, positioned below the word "android".



apple
ios

ANDROID



1. AOSP (Android Open Source Project)
2. Linux kernel
3. GMS (Google Mobile Services)
4. Apps written in Java, Kotlin, others
5. Linux, but not as we know it

iOS

1. Darwin (BSD) kernel
2. Cocoa Touch user interface
3. Apps written in Objective-C, Swift
4. Closed source, closed ecosystem



LINUX

1. Mobian (Debian + Phosh)
2. postmarketOS (Alpine Linux + Plasma Mobile)
3. Ubuntu Touch (Ubuntu + Lomiri)
4. Nemo Mobile (Manjaro + Nemo)
5. Sailfish OS (Mer + Silica)



ubports



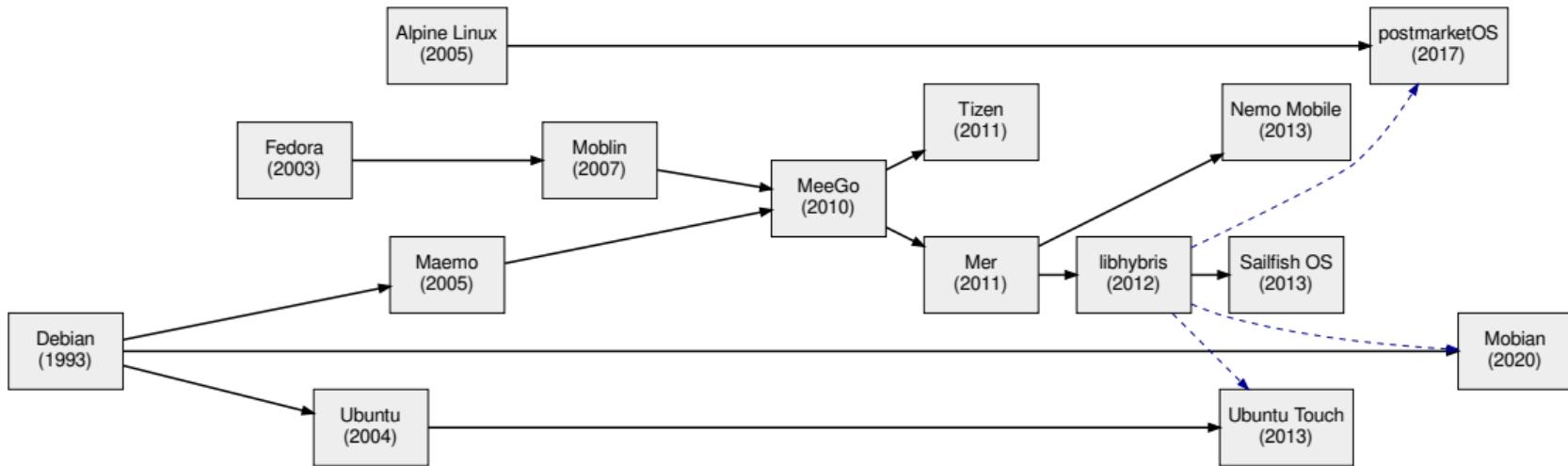
SAILFISH OS



WHY ARE THEY INTERESTING?

1. Linux, glibc, GNU Core Utils, systemd
2. Open Source
3. Open development models
4. Open ecosystems
5. Community-oriented business models
6. Most Linux software just compiles and runs

LINUX ON MOBILE HISTORY



MAEMO 4.1



MAEMO 5



SALFISH OS 2.2





SOKOS

partio helsingia
LNUA

HENRY'S PUB

GRILLI

PARTIO
KÄRKELE

EURO
RENT
www.eurorent.fi

UBUNTU TOUCH 16.04



POSTMARKETOS 22.12.2



SAILFISH OS 4.5



SAILFISH OS STACK

1. Android drivers
2. Libhybris and Libgbinder
3. Linux kernel 4.19.248
4. glibc
5. systemd
6. busybox

```
[root@raspberrypi ~]# emerge -s
* Emerging Start
* Emerging libhybris
* Emerging libgbinder
* Emerging glibc
* Emerging busybox
```

Information for package libhybris:

Name	Version	Architecture	Description
libhybris	0.0.5.48-1.5.3.jolla	armv7a	utilite Android-based W/A abstractions on glibc systems

Information for package libgbinder:

Name	Version	Architecture	Description
libgbinder	1.3.30-6.3.jolla	armv7a	utilite Android-based W/A abstractions on glibc systems

Information for package glibc:

Name	Version	Architecture	Description
glibc	2.28-r101-1.8.5.jolla	armv7a	The C library contains standard libraries which are used by multiple programs on the system. In order to save disk space by memory, as well as to make upgrading easier, certain system code is shared between multiple programs. This means that the C library contains the most important sets of shared libraries: the standard C library and the C standard library. Without these two libraries, a Linux system will not function.

Information for package busybox:

Name	Version	Architecture	Description
busybox	1.34.1-r101-1.7.4.jolla	armv7a	busybox is a single binary which includes versions of a large number of system commands, including a shell.



SAILFISH OS STACK



7. Qt middleware
8. Wayland + Lipstick compositor
9. Silica user interface
10. Lipstick launcher
11. Gecko-based Web browser
12. Android App Support

LIBHYBRIS AND LIBGBINDER

Libhybris

1. Allows AOSP drivers to be used with glibc Linux
2. Dynamic loading of Android libraries
3. Overrides bionic symbols with glibc symbols

Libgbinder

1. Android Binder Protocol
2. HAL Interface definition language
3. Switching from linking to Binder
4. Modems switched from socket to Binder in Android 8

DBUS

Inter-process communication

1. Object-oriented and typed
2. System bus, per-login session bus
3. Peer-to-peer or bus-oriented
4. Properties, method calls, signals, introspection

```
1 class FlashlightDBusAdaptor: public QDBusAbstractAdaptor
2 {
3     Q_OBJECT
4     Q_CLASSINFO("D-Bus Interface",
5                 "com.jolla.settings.system.flashlight")
6
7     public:
8     Q_PROPERTY(bool flashlightOn READ flashlightOn)
9
10    FlashlightDBusAdaptor(QObject *parent);
11    bool flashlightOn() const;
12
13    public slots:
14        bool toggleFlashlight();
15
16    signals:
17        void flashlightOnChanged();
18};
```

```
1 dbus-send --session --type="method_call" --print-reply \
2   --dest="com.jolla.settings.system.flashlight" \
3   "/com/jolla/settings/system/flashlight" \
4   "com.jolla.settings.system.flashlight.toggleFlashlight"
```

QT

1. Middleware libraries
2. User Interface toolkit
3. Cross-platform C++, PyOtherSide
4. Meta-Object Compiler
5. QObject model

```
1 #include <QObject>
2
3 class Example : public QObject
4 {
5     Q_OBJECT
6
7     Q_PROPERTY(bool selected READ selected WRITE setSelected
8 NOTIFY selectedChanged)
8 public:
9     explicit Example(QObject *parent = nullptr)
10        : QObject(parent)
11        , m_selected(false)
12 {}
13
14     bool selected() const { return m_selected; }
15
16     void setSelected(bool selected) {
17         if (m_selected != selected) {
18             m_selected = selected;
19             emit selectedChanged();
20         }
21     }
22
23 signals:
24     void selectedChanged();
25
26 private:
27     bool m_selected;
28 };
```

QML

1. Declarative user interface language
2. *Components, properties and functions*
3. Property changes trigger recalculation
4. Imperative JavaScript functions
5. Rendered components positioned using anchors

```
1 Page {
2     property int count: 0
3
4     Example {
5         id: example
6         selected: toggle.checked
7     }
8
9     Timer {
10        id: countdown; interval: 1000; repeat: true
11        onTriggered: if (count > 0) count -= 1
12        running: example.selected == true
13        onRunningChanged: count = 10
14    }
15
16    Column {
17        id: column; anchors.fill: parent
18        PageHeader { title: qsTr("REG Tech Talk Example") }
19        TextSwitch {
20            id: toggle
21            text: qsTr("Activated")
22        }
23
24        Label {
25            x: Theme.horizontalPageMargin
26            text: example.selected
27            ? qsTr("This message will self destruct in
28                %1 seconds").arg(count)
29            : qsTr("Disarmed")
30        }
31    }
32 }
```



SAILFISH OS

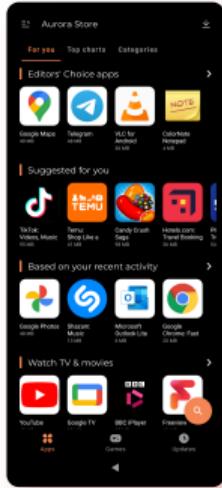
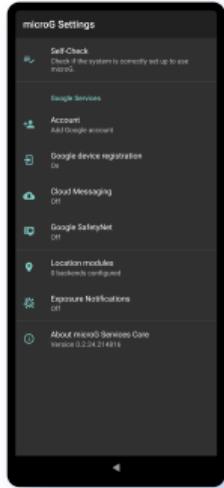
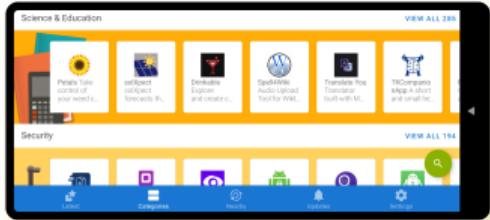
HACKING THE HOMESCREEN

1. QML interface is highly hackable
2. A simple example:

/usr/share/lipstick-jolla-home-qt5/statusarea/StatusArea.qml

```
141     Label {
142         id: name
143         text: "David's phone"
144         anchors.verticalCenter: parent.verticalCenter
145     }
```

ANDROID APP SUPPORT



FURTHER INFO

Sailfish OS <https://sailfishos.org>

Ubuntu Touch <https://ubports.com>

postmarketOS <https://postmarketos.org>

Mobian <https://mobian-project.org>

Nemo Mobile <https://nemomobile.net>

Slides source <https://github.com/llewelld/reg-tech-talk-linux>

IS IT OPEN SOURCE?

Closed drivers, everything else open

1. Mobian
2. postmarketOS
3. Ubuntu Touch
4. Nemo Mobile

Sailfish OS

1. Closed drivers
2. Linux kernel open
3. Middleware, Qt open
4. User interface closed
5. Jolla apps closed
6. Android App Support closed

EXCITING COMMUNITY PROJECTS

Lots of neat stuff...

1. Flatpack support
2. Sailfish on x86
3. AsteroidOS
4. SDK Rust support



SUPPORTED DEVICES

Distribution	Official	Community	More info
Sailfish OS	Xperia X, XA2, 10, 10 II, 10 III, Gemini	PinePhone, Fairphone 2, Galaxy Note 4, F(x)Tec Pro, Volla, plus at least 20 others	forum.sailfishos.org/t/14081
Mobian	PinePhone, Librem 5, OnePlus 6/6T, Pocophone F1	Fairphone 4, SHIFT6mq, Mi Mix 2S	wiki.mobian.org/?id=devices
postmarketOS	PinePhone, Librem 5	Fairphone 4, OnePlus 6/6T, Galaxy A3/A5/E7/Tab, Pocophone F1, Mi Note 2, plus at least 20 more	postmarketos.org/download
Ubuntu Touch	Volla, Fairphone 2, Nexus 5, OnePlus One, PinePhone	Pixel 3a, Poco X3, Mi A2, BQ Aquaris M10, Asus Zenfone Max Pro M1, Xperia X	devices.ubuntu-touch.io
Nemo Mobile	PinePhone, Volla	–	nemomobile.net/devices



SAILFISH OS

PYTORCH LIGHTNING

```
1 zypper install gcc python3-devel
2 python3 -m venv venv
3 . ./venv/bin/activate
4 python3 -m pip install torch lightning torchvision
5 python3 example.py

1 # See: https://lightning.ai/docs/pytorch/stable/starter/introduction.html
2
3 import os
4 from torch import optim, nn, utils, Tensor
5 from torchvision.datasets import MNIST
6 from torchvision.transforms import ToTensor
7 import lightning.pytorch as pl
8
9 # define any number of nn.Modules (or use your current ones)
10 encoder = nn.Sequential(nn.Linear(28 * 28, 64), nn.ReLU(), nn.
11     .Linear(64, 3))
12 decoder = nn.Sequential(nn.Linear(3, 64), nn.ReLU(), nn.
13     Linear(64, 28 * 28))
14
15 # define the LightningModule
16 class LitAutoEncoder(pl.LightningModule):
17     def __init__(self, encoder, decoder):
18         super().__init__()
19         self.encoder = encoder
20         self.decoder = decoder
```

```
20     def training_step(self, batch, batch_idx):
21         # training_step defines the train loop.
22         # it is independent of forward
23         x, y = batch
24         x = x.view(x.size(0), -1)
25         z = self.encoder(x)
26         x_hat = self.decoder(z)
27         loss = nn.functional.mse_loss(x_hat, x)
28         # Logging to TensorBoard (if installed) by default
29         self.log("train_loss", loss)
30         return loss
31
32     def configure_optimizers(self):
33         optimizer = optim.Adam(self.parameters(), lr=1e-3)
34         return optimizer
35
36     # init the autoencoder
37     autoencoder = LitAutoEncoder(encoder, decoder)
38
39     # setup data
40     dataset = MNIST(os.getcwd(), download=True, transform=
41                     ToTensor())
42     train_loader = utils.data.DataLoader(dataset, num_workers=4)
43
44     # train the model (hint: here are some helpful Trainer
45     # arguments for rapid idea iteration)
46     trainer = pl.Trainer(limit_train_batches=100, max_epochs=1)
47     trainer.fit(model=autoencoder, train_dataloaders=train_loader
48 )
```



SAILFISH OS

JUPYTER LAB

```
1 python3 -m venv venv-jupyter
2 . ./venv-jupyter/bin/activate
3 pip install jupyter ipykernel matplotlib
4 python3 -m ipykernel install --user --name=venv-jupyter
5 jupyter notebook password
6 jupyter notebook --ip 10.0.0.42 --port 8888
```

The screenshot shows the Jupyter Notebook interface. In the top right corner, there is a Python logo icon. Below it, the text "jupyter notebook" and "Logout". On the left, there's a toolbar with icons for file operations like new, open, save, and run. The main area has two code cells labeled In [5] and In [6].

In [5]:

```
import numpy as np

def sample_spherical(npoints, ndim=3):
    vec = np.random.randn(ndim, npoint)
    vec /= np.linalg.norm(vec, axis=0)
    return vec
```

In [6]:

```
from matplotlib import pyplot as plt
from mpl_toolkits.mplot3d import axes3d

phi = np.linspace(0, np.pi, 20)
theta = np.linspace(0, 2 * np.pi, 40)
x = np.outer(np.sin(theta), np.cos(phi))
y = np.outer(np.sin(theta), np.sin(phi))
z = np.outer(np.cos(theta), np.ones_like(phi))

xi, yi, zi = sample_spherical(100)

fig, ax = plt.subplots(1, 1, subplot_kw=dict(projection='3d'))
ax.plot_wireframe(x, y, z, color='k')
ax.scatter(xi, yi, zi, s=100, c='r')
plt.show()
```

Below the code cells is a 3D scatter plot of a sphere. The sphere is centered at the origin (0,0,0) and has a radius of approximately 1.0. The surface is covered with numerous small red dots, which are the points generated by the function in In [5]. The plot is set against a grid.