State of Mobile Linux

Juha-Matti Liukkonen, Jan 5, 2011

Contents

Why is this interesting in a Qt course?

Rationale

Advances in technology make computers mob

Low-power processors, displays, wireless network chipsets, ...

Laptops outsell desktop computers

iSuppli, Dec 2008

High-end smartphones = mobile computers

Need to know how to make software function well in a mobile device

Nokia terminology

Qt is big part of Symbian & Maemo/MeeGo ...

Developing software for mobiles

Android smartphones

Eclipse, Java

Symbian smartphones

NetBeans / Eclipse, Java ME

Qt Creator, C/C++

Maemo / MeeGo smartphones

Qt Creator, C/C++

In desktop/server computing:
Java :== server
C/C++ :== desktop

Qt was initially developed for desktop applications.

Mobile devices today are more powerful than the desktops 10 years ago.

Of particular interest in this course.

The elephant in the room

In 2007, Apple change the mobile world with the iPhone

Touch user interface, excellent developer tools, seamless services integration, …

Modern operating system, shared with iPod and Mac product lines

Caught “industry regulars” with their pants down

Nokia, Google, Samsung, et al – what choice do they have? Linux!

We don’t talk about the iPhone here.

iPad “killed the netbook”

In 2010, Apple introduced another mobile game changer

iPad = basically, a scaled-up iPhone with a 10” touch screen

Bigger case = can fit bigger battery, for ~10 hours of intensive use

Apple, 2010

7,5 million sold as of Sep 30, 2010

Phenomenal netbook sales growth fizzled

NPD, Morgan Stanley Research, 2010

App ecosystems

Powerful mobile computers can run variety of software

Dynamic availability of applications to provide added value over device lifetime

Devices become multi-purpose and adaptable

Voice call functionality a secondary feature

New software design challenges

Nokia was ahead of its time with the Communicator concept back in 1996.

New user interaction models

Preservation of battery

Mobile device constraints

Mobility = situations change

May lose network coverage

May run out of battery

> Network connections error out.
> Your app dies while writing to a file.
> Your app is frozen to let a call through.

Software needs to adapt to situation at hand

Mobility = limited resources

CPUs, GPUs not as fast as on desktop systems

Smaller screens, different input devices

Limited amount of battery power

Cross development

Cannot compile software in the target device

Not enough memory, disk space, CPU power

Poor input/output devices for development

Must use a cross-compile environment

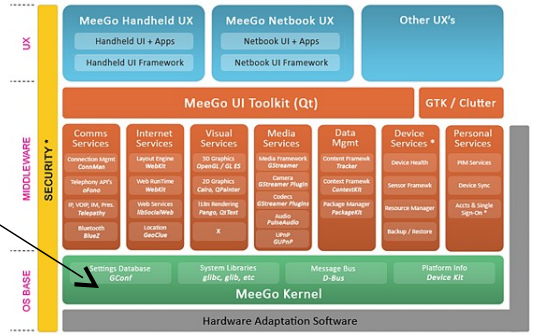
SDK = Software Development Kit

Build software in e.g. QtCreator, compile with SDK tools, install & run in the target device

Debugging software in target is often a bit tricky. Most SDKs come with a device emulator.

Mobile Linux distributions

Linux Distribution



Linux is the operating system kernel

Deals with hardware abstraction

A *distribution* is a managed collection of software, including the kernel

Device drivers, middleware, user applications

Comes with distributor-defined default settings and applications

Often optimized for specific use(s)

E.g. Ubuntu, Red Hat Enterprise Linux, Maemo

Mobile Linux distributions

Maemo

Nokia’s Linux distribution for Internet tablets and high-end smartphones

Powers the N770, N800, N810, N900

Android

Google’s Linux distribution for Internet tablets and smartphones

Powers many HTC devices, Nexus One, etc.

Mobile Linux distributions

OpenEmbedded

Open source project

Best suited for custom adaptations to very small devices

MeeGo

New kid on the block

Combines Intel’s Moblin netbook Linux and Nokia’s Maemo Linux

First MeeGo devices out in fall 2010

Android details

Uses custom Linux kernel

Google maintains a set of Android patches

Applications developed using Java

Google’s custom Dalvik Java VM

6 versions in active use

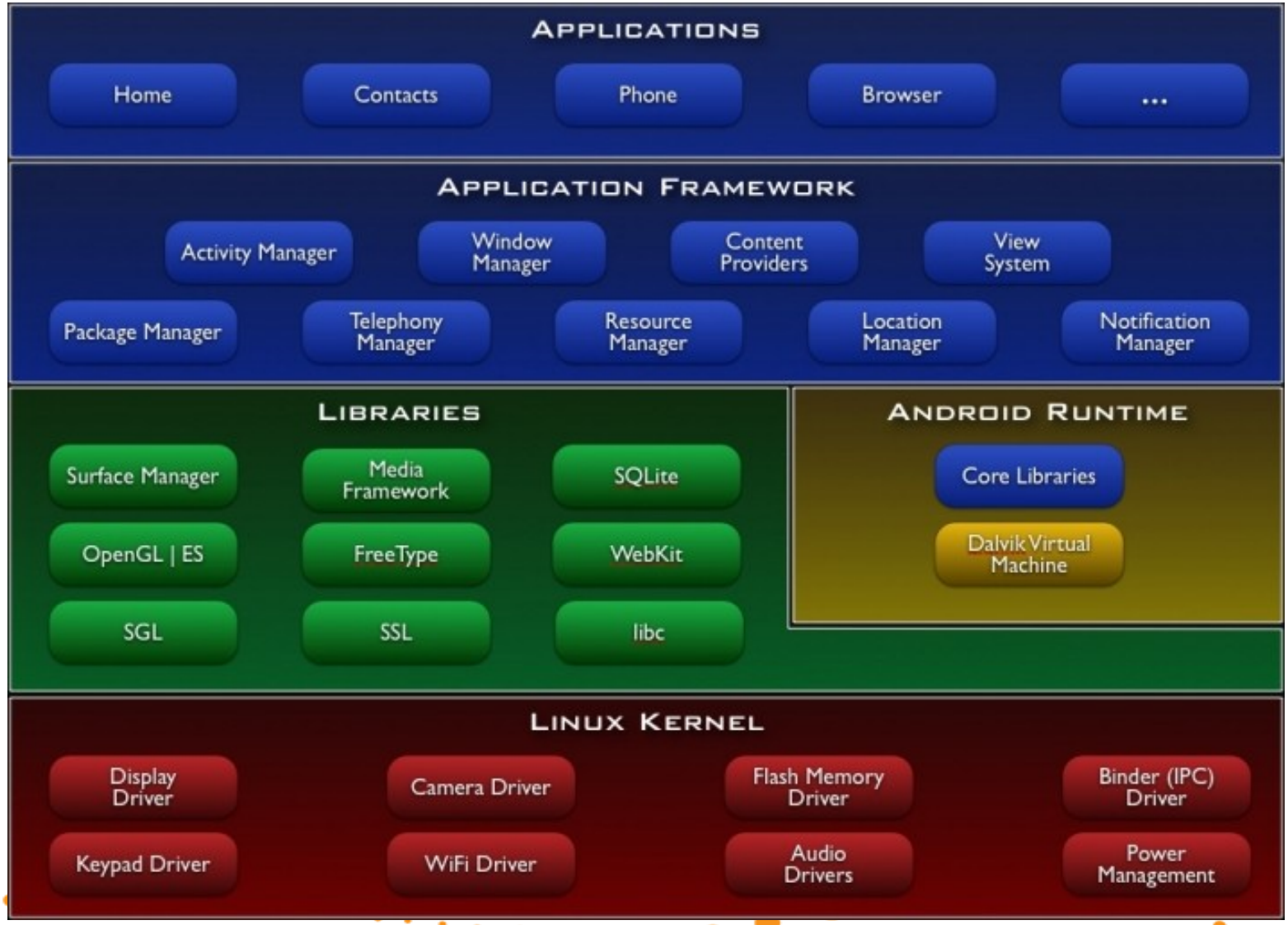
1.5, 1.6, 2.0, 2.1, 2.2 and now 2.3

Used in various smartphones by HTC, Google, Motorola, LG, etc.

There is also a Native Development Kit (NDK) for building native Linux applications.

The devices have a bit different resolutions and feature sets.

Android architecture



Android points of interest

Custom C library

C library = system calls (interface to kernel),
POSIX & ANSI standard library routines

Linux standard is glibc, which is a bit bloated

Android has a stripped down libc

Compatibility issues for generic Linux code

Custom application installation

Apps bundled into .apk “Android packages”

Android points of interest

Programming model

Activity

- Impements an application view

Service

- Background program with no UI

Broadcast receiver

- Listens for e.g. battery notifications

Content provider

- Shares data from an app

Android and Qt

Project Lighthouse = Qt for Android

Project ongoing… not ready for prime time yet

Some limitations of Android Native SDK (NDK) cause problems

Should eventually allow Qt to be the universal (mobile) Linux toolkit!

Google is not very supportive ;-)

Maemo details

Uses standard Linux kernel

Applications developed using Qt, C/C++

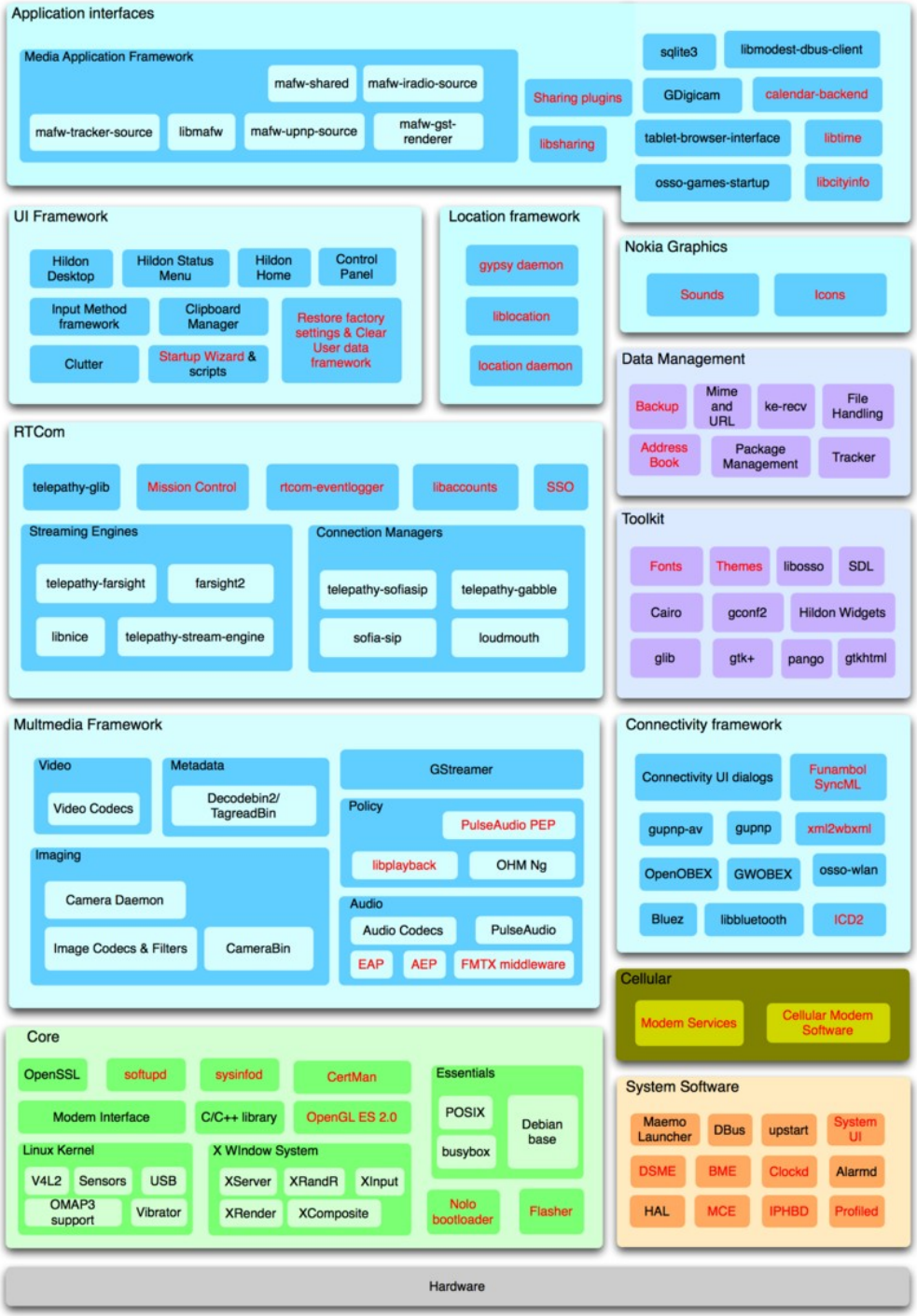
Maemo 4 & 5 are GTK based, but even there Qt is the recommended development toolkit

Maemo 4 used in Nokia N800 tablet (deprecated)

Maemo 5 used in Nokia N900 smartphone

Maemo 6 this fall -> MeeGo

Maemo 5 architecture



Maemo points of interest

Very regular Linux in most ways

Debian based, uses dpkg & apt-get

Uses glibc, gstreamer, X.org, etc.

User interface based on Hildon/GTK+

Clutter backend for fancy effects

Qt natively supported

Nokia SDK = QtCreator + SDKs for Symbian/Maemo

Qt apps for Maemo 5 have the Maemo look & feel, support touch input, etc.

MeeGo details

Uses standard Linux kernel

Applications developed using Qt, C/C++

Replaces both Moblin from Intel, and Maemo from Nokia

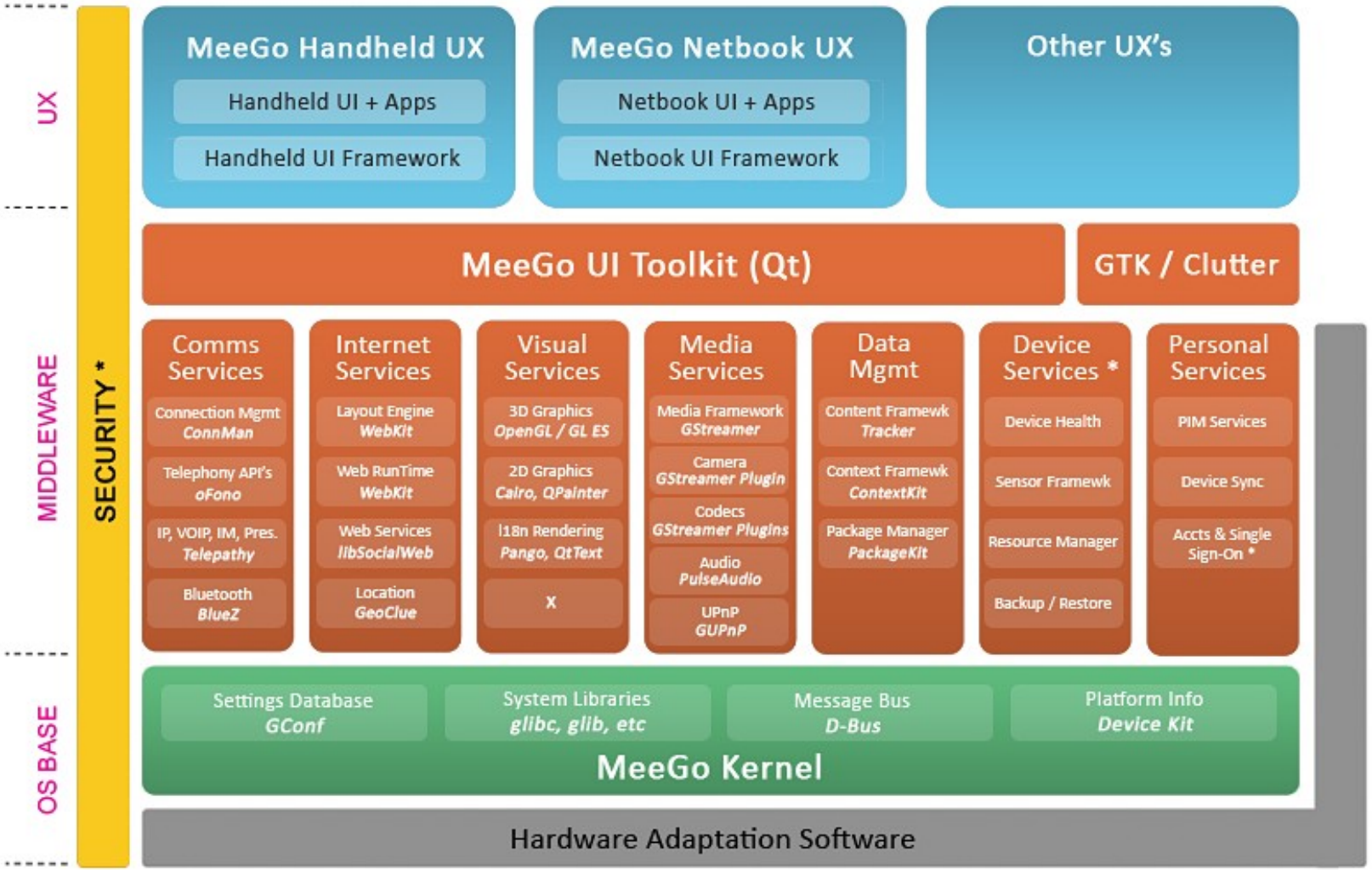
Some products released already

Alpha quality 1.0 release May 2010

Beta quality 1.1 release Oct 2010

Product quality 1.2 release Apr 2011

MeeGo architecture



MeeGo points of interest

Mostly regular Linux

Glibc, gstreamer, ALSA, etc.

Not based on any existing distribution, but uses rpm & zypper for package management

User interface modules separated from base platform

Different user interaction models for different use scenarios

Qt is the primary application interface

MeeGo points of interest

UX modules

Handheld: touchscreen (meegotouch toolkit on top of Qt)

Netbook: keyboard/mouse

Connected TV: remote control

In-Vehicle Information: touchscreen, joystick

Reference applications for each UX model

System vendors can customize as needed

MeeGo points of interest

Stable API

Any MeeGo application can run on any MeeGo certified system

Main part of API is Qt (Core, Gui, Mobility, …)

Also: gstreamer, sqlite, ALSA, D-BUS interfaces to various frameworks, etc.

Goal is to encourage an App Store ecosystem rivaling Apple

Why MeeGo is interesting to us

Only credible challenger to Android

Backed by Nokia -> direct impact to Finnish software development scene

You can participate in building MeeGo: go to meego.com and become active!

Innovative architectural solutions

Aims to become the “industry standard” Linux for modern embedded systems

Will drive Qt development in mobile space

Participate to MeeGo!

<symbio>

Go to meego.com and register as a developer

Participate in community working groups

Discuss in #meego at freenode

Contribute code, documentation, tests

Gain reputation, become a component maintainer

Steering group meetings in #meego-meeting

Help us create the future of mobile Linux!

Developing software for mobile Linux

Software design considerations

Mobile environment constraints

Limited battery power

Limited CPU power

Limited screen size

Changing situations

Not that difficult to work with, once you know what to avoid

Mobile optimized software runs *fast* on higher end hardware

Design your app for a CPU from 1990, graphics from 2005, but use modern tools and techniques.

Limited battery power

Typical smartphone battery is around 1300-1500 mAh

An ARM Cortex A8 @ 600 MHz draws 300 mW - an Intel XEON @ 3 GHz draws 130W

Software must do as little as possible

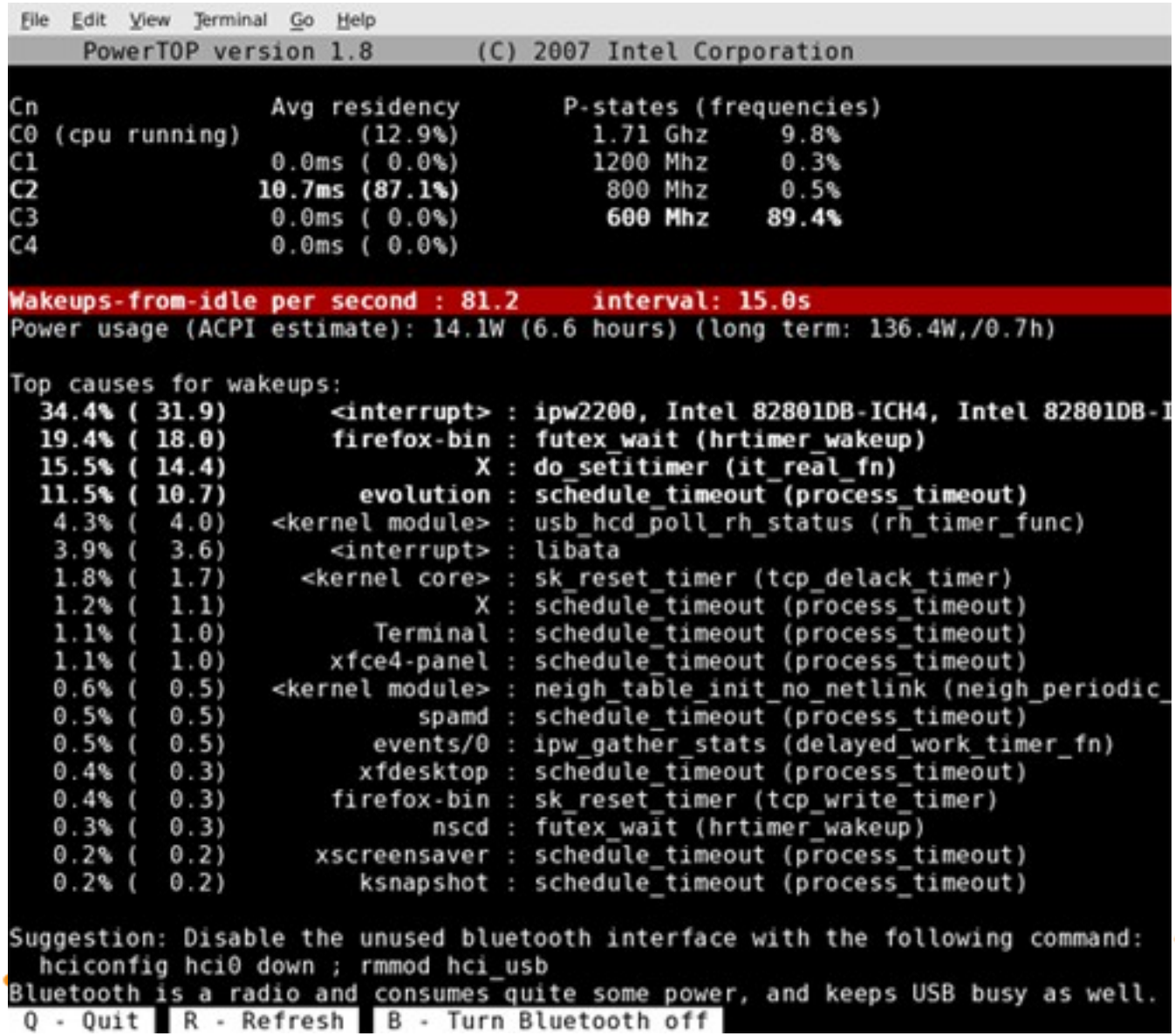
Must not poll for network traffic, user input, ambient light sensors, ...

Applications must become *context aware.*

Must not update the screen when in background

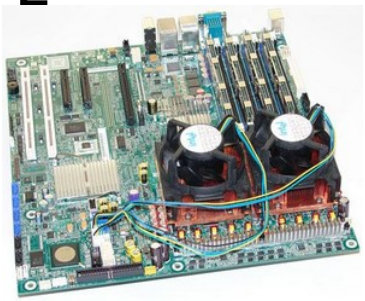
Use platform services for notifications

Tools: powertop



Limited CPU power

Desktop machines today use 2-4 CPU cores of 2-3 GHz each (4x3 GHz)



Server machines have 3-4x that

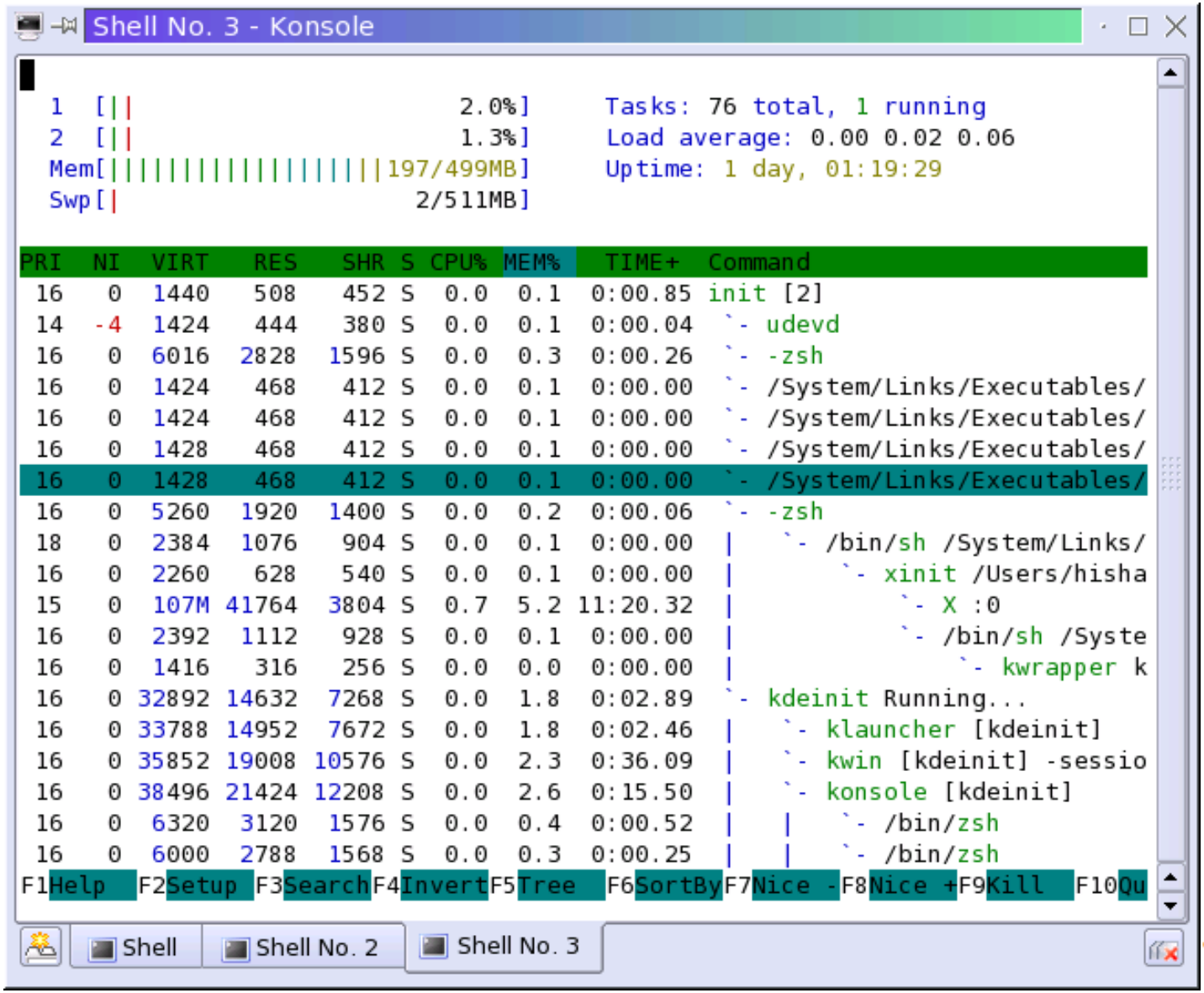
Netbooks and smartphones have 1-2 cores of 0.5-1 GHz

Level of parallelization is very different

Maximum throughput is very different

Efficient algorithms work on smartphone level CPUs, and scream on high-end computers

Tools: htop


Shell No. 3 - Konsole
1 [|| 2.0%] Tasks: 76 total, 1 running
2 [|| 1.3%] Load average: 0.00 0.02 0.06
Mem[||||||||||||||||||197/499MB] Uptime: 1 day, 01:19:29
Swp[| 2/511MB]
PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
16 0 1440 508 452 S 0.0 0.1 0:00.85 init [2]
14 -4 1424 444 380 S 0.0 0.1 0:00.04 `- udevd
16 0 6016 2828 1596 S 0.0 0.3 0:00.26 `- -zsh
16 0 1424 468 412 S 0.0 0.1 0:00.00 `- /System/Links/Executables/
16 0 1424 468 412 S 0.0 0.1 0:00.00 `- /System/Links/Executables/
16 0 1428 468 412 S 0.0 0.1 0:00.00 `- /System/Links/Executables/
16 0 1428 468 412 S 0.0 0.1 0:00.00 `- /System/Links/Executables/
16 0 5260 1920 1400 S 0.0 0.2 0:00.06 `- -zsh
18 0 2384 1076 904 S 0.0 0.1 0:00.00 | `- /bin/sh /System/Links/
16 0 2260 628 540 S 0.0 0.1 0:00.00 | `- xinit /Users/hisha
15 0 107M 41764 3804 S 0.7 5.2 11:20.32 | `- X :0
16 0 2392 1112 928 S 0.0 0.1 0:00.00 | `- /bin/sh /Syste
16 0 1416 316 256 S 0.0 0.0 0:00.00 | `- kwrapper k
16 0 32892 14632 7268 S 0.0 1.8 0:02.89 `- kdeinit Running...
16 0 33788 14952 7672 S 0.0 1.8 0:02.46 | `- klauncher [kdeinit]
16 0 35852 19008 10576 S 0.0 2.3 0:36.09 | `- kwin [kdeinit] -sessio
16 0 38496 21424 12208 S 0.0 2.6 0:15.50 | `- konsole [kdeinit]
16 0 6320 3120 1576 S 0.0 0.4 0:00.52 | | `- /bin/zsh
16 0 6000 2788 1568 S 0.0 0.3 0:00.25 | | `- /bin/zsh
F1Help F2Setup F3SearchF4InvertF5Tree F6SortByF7Nice -F8Nice +F9Kill F10Qu
Shell
Shell No. 2
Shell No. 3

Limited screen size

Laptops may have 17”, 1920x1200 screens

Netbooks have 7-13”, max 1280x800

Smartphones have 3-4”, max 860x480

Designing a scaling application UI is hard

Dialogs designed for 860x480 may look tiny on 1920x1200

Dialogs designed for 1920x1200 may simply not fit in 860x480

Also, what is the input mechanism?

Changing situations

May lose network coverage

Applications must degrade gracefully

May start to run out of power

Must absolutely minimize what apps do

Must survive power outages gracefully

Incoming phone call

Applications must yield immediately to allow the high priority task to run

You always *do* check for error values from API calls, right?

Database transactions and journaling file systems are your friends.

User interaction

Device rotation

Mobile devices often have accelerometers – can tell whether it is in landscape or portrait mode

Applications should register for orientation change notifications and re-layout accordingly

Input methods

Hardware keyboard, virtual keyboard, finger input, Bluetooth keyboards and mice, ...

Extra controls such as microphone buttons

Catching rotation on Maemo

MyReceiverClass inherits *QObject*, and implements slot *orientationChanged().*

```
MyReceiverClass * receiver = new MyReceiverClass(this);
QDBusConnection systemBus = QDBusConnection::systemBus();
systemBus.connect("com.nokia.mce",
                  "/com/nokia/mce/signal",
                  "com.nokia.mce.signal",
                  "sig_device_orientation_ind",
                  receiver, SLOT(orientationChanged(QString)));
```

DBus is a message passing system.

MCE is the Mode Control Entity (a system process) in Maemo.

When MCE sends the *sig_device_orientation_ind* via the system message bus, your object's slot *orientationChanged()* will get called with argument "portrait" or "landscape".

Memory management

Memory leaks cause device to reboot

User will not like this! (user = *you*)

Careful weeding out of dynamic memory allocation problems is needed

Use QObjects, tools such as valgrind

Limited amount of memory in device

Swapping is slow and expensive power-wise

Use as little memory as you can, free memory when not needed

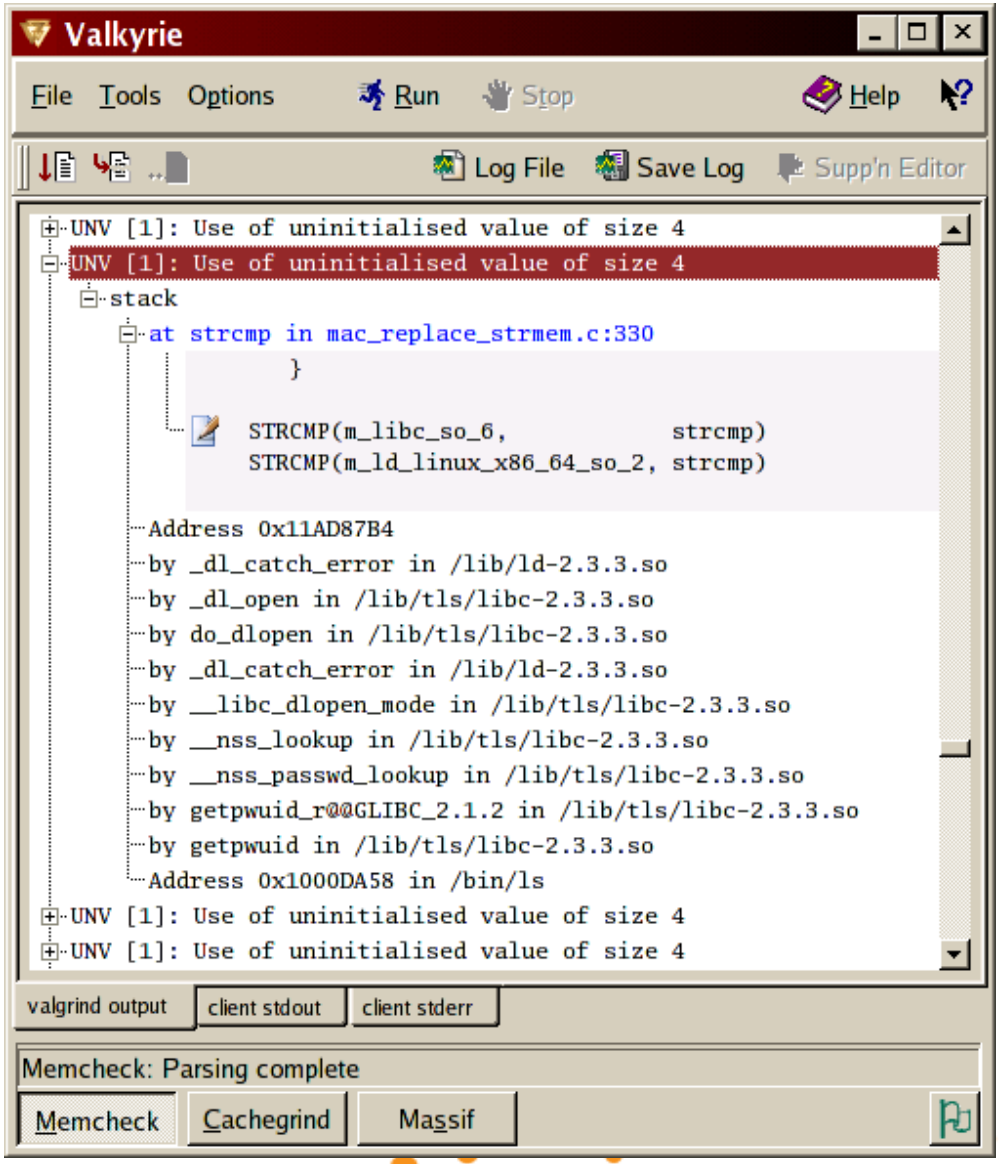
Arguably, Java is better than C/C++ in this regard as it handles memory deallocation automatically.

Tool: valgrind

Supports x86, older ARM chips

Full ARMv7 support coming soon

Multiple GUI front-ends



Multitasking

Running multiple applications simultaneously requires special care

You run out of memory, and device reboots

Background apps eat all your CPU, and you can’t answer phone calls

iPhone OS just tells your app it’s about to be killed

Your app must save state information to disk, so that it can resume smoothly when restarted

This is clever, because only one app at a time is in memory.

Multitasking

Android does automatic suspend/resume

Stores idle app state on disk automatically

Reloads app state when app is resumed, which causes occasional stalls

Maemo/MeeGo does regular Linux multitasking

cgroups to prioritize process groups in memory

Uses swap to extend physical memory, which causes occasional stalls

Android 2.2 has a task manager, where you can kill idle apps by hand.

User interface design

Mobile design patterns

Mobilize, Don’t Miniaturize

The Carry Principle

Context Sensitivity

New research coming out all the time

Including 3D interfaces in 1-2 years

Tricky to combine 3D and touch…

Multiplatform applications

Qt is supported “everywhere”

Encourages creation of multiplatform applications

But: even if the core application works everywhere, the user interface may not

#ifdef statements in code to instantiate different UI code for different UX environments?

QtMobility APIs to query system features

Web widgets

WebKit is the standard web runtime (WRT)

Provided by Android, Symbian, Maemo/MeeGo

Differences in JIT support

Platform service interfaces visible in JavaScript

Applications as widgets run in a WRT process context

Similar considerations as for native applications

Often possible to embed to native applications, sometimes just use in homescreen etc.


symbio
SERIOUS ABOUT SOFTWARE