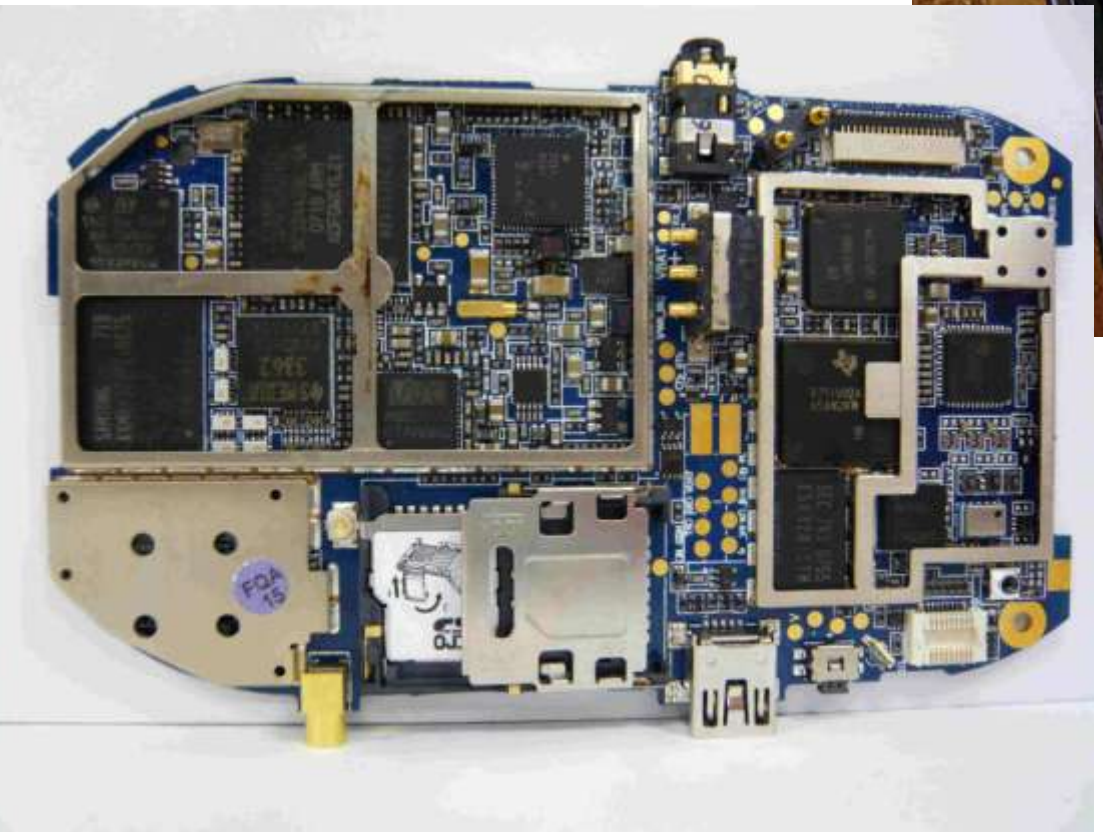
Systemy operacyjne na urządzenia mobilne

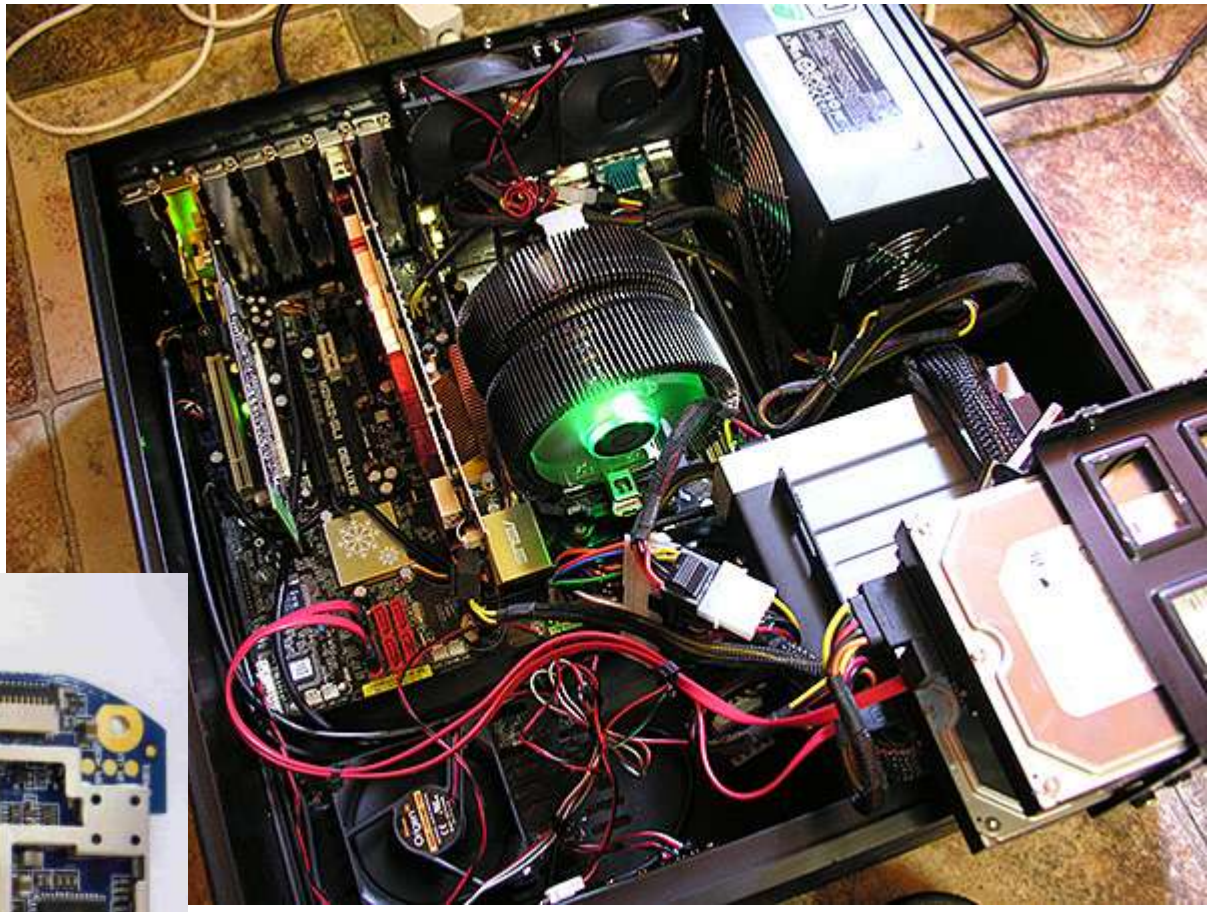
Plan

- Urządzenia mobilne – ogólnie
- Przegląd systemów
  - Windows Mobile
  - Linux
  - Symbian
  - Google Android

Różnice?

mobile
objętość ~50ml
moc ~ 0.5 W





PC
objętość ~100l
moc ~ 300 W

Rodzaje urządzeń przenośnych

- PDA (Personal Digital Assistant)

- Tablet PC

- Smartphone

	
CPU 412 MHz RISC, 32-bit	CPU 2 GHz CISC, 64-bit
128 MB RAM	2 GB RAM
Wbudowany moduł GSM, Bluetooth, GPS, WiFi	Karty rozszerzeń

Wyraźna różnica w parametrach sprzętu, ale z drugiej strony...

Windows 95 działał nawet na procesorze 486 przy 4 MB RAM

Kluczową różnicą stanowią raczej wymagania

(agresywne oszczędzanie energii, podsystemy real-time, ...)

niż wydajność sprzętu

Czego chce użytkownik?

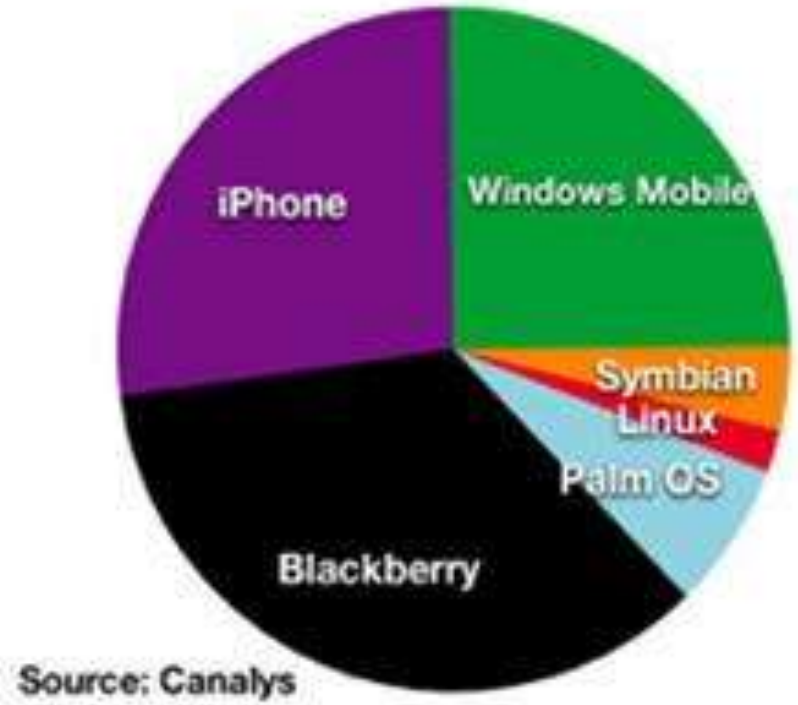
1. Niezawodność
   a) Stabilna praca
   b) Usługi czasu rzeczywistego

2. Funkcjonalność

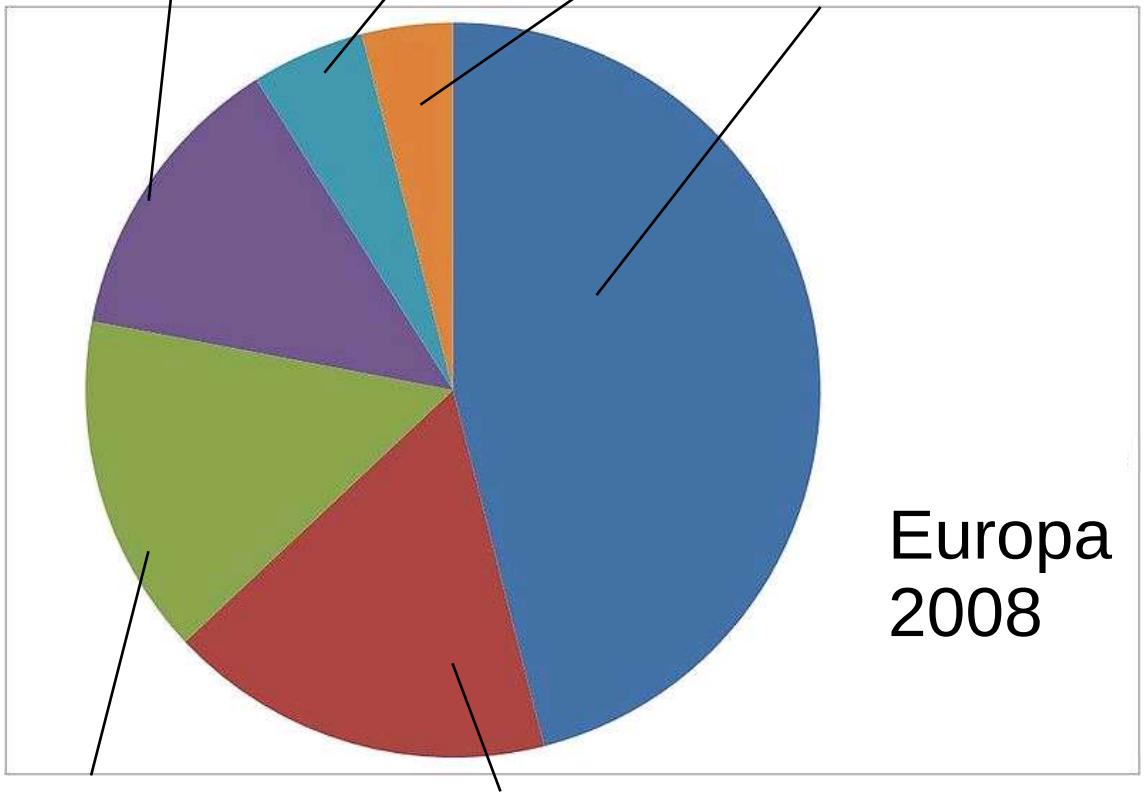
3. Bezpieczeństwo
   a) Zagrożenia analogiczne jak w PC
   b) Środowisko stosunkowo homogeniczne

Popularność poszczególnych systemów

(na podstawie liczby sprzedanych urządzeń)



USA 2007

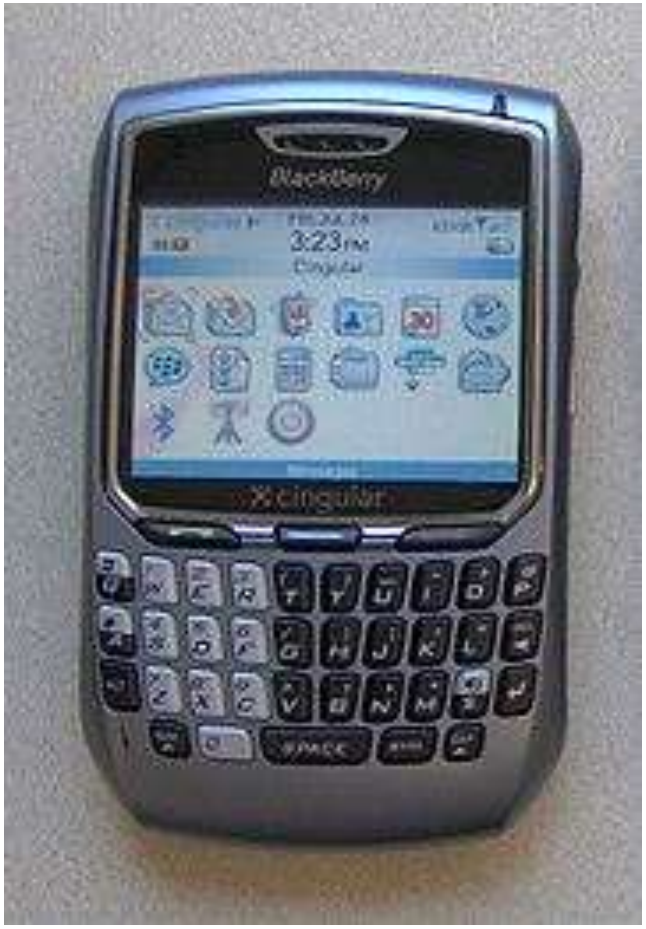


Europa 2008

BlackBerry

Urządzenie popularne w kręgach biznesowych, zwłaszcza w USA.
Dedykowany system operacyjny producenta (bardzo konkretne sterowniki, obsługa WAP na poziomie jądra, itp.)

- „Mobile e-mail gold standard”
- Produkt firmy
  Research in Motion (RIM)
- Wymaga punktu
  dostępowego (tzw. *apn*)
- *e-mail push* (jak w IMAP)



Windows Mobile™

Windows Mobile od strony użytkownika

System operacyjny + podstawowe aplikacje = platforma

W smartphonie można zainstalować inne aplikacji niż domyślne, ale kto to robi?

- Office Mobile
- Outlook Mobile
- IE Mobile
- Windows Media Player
- Internet Connection Sharing
- ActiveSync

Wersje Windows Mobile 6

- Windows Mobile 6 Standard
  (GSM, bez ekranu dotykowego)
- Windows Mobile 6 Professional
  (GSM, z ekranem dotykowym)
- Windows Mobile 6 Classic
  (bez GSM)

Rodowód Windows Mobile

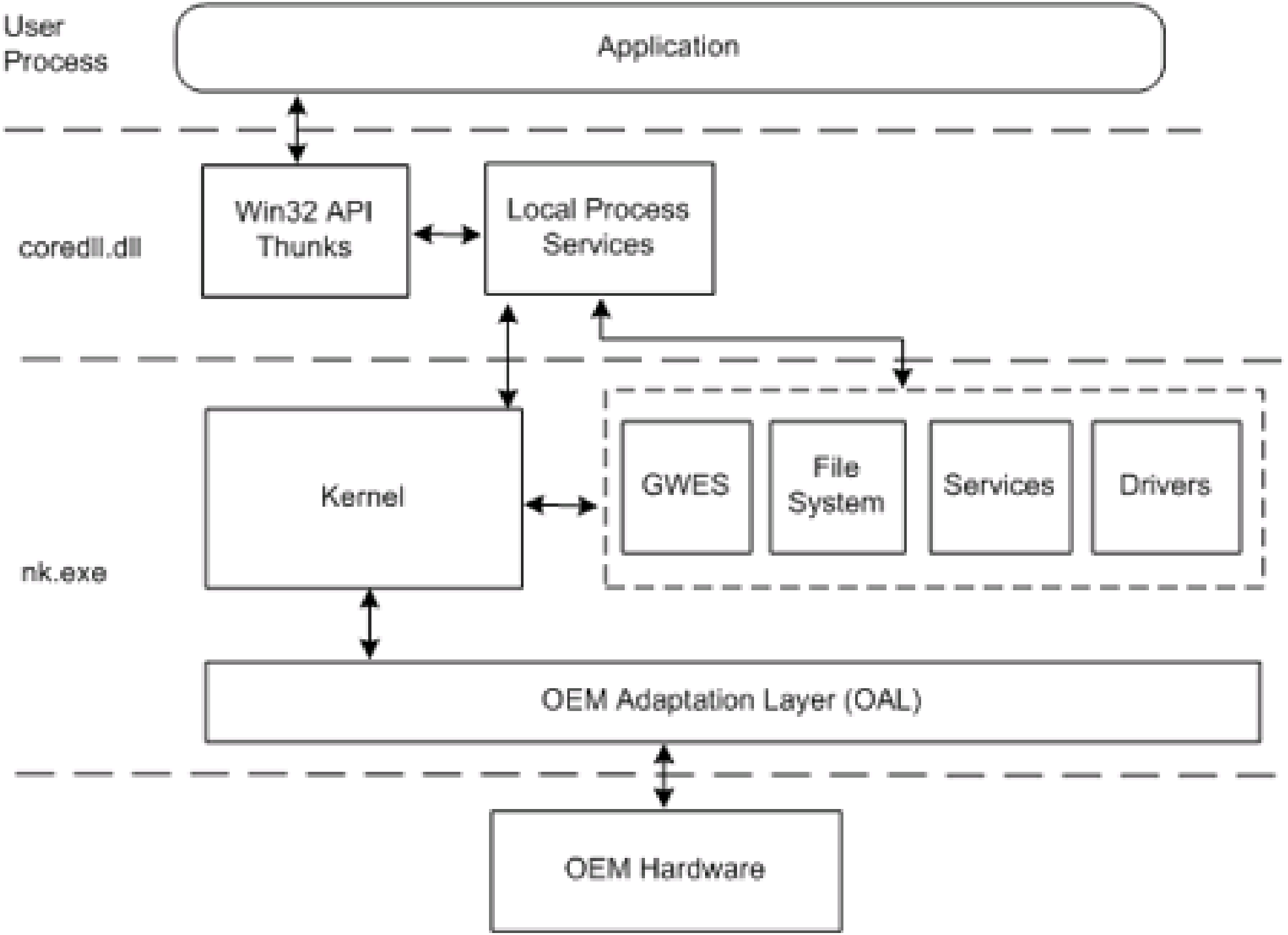
- Pocket PC 2000
- Pocket PC 2002
- Windows Mobile 2003
- Windows Mobile 5
- Windows Mobile 6.1
- Windows Mobile 7
  (II połowa 2009)

- Pegasus (1995)

- Windows CE 1 (1996)

Windows na urządzenia przenośne

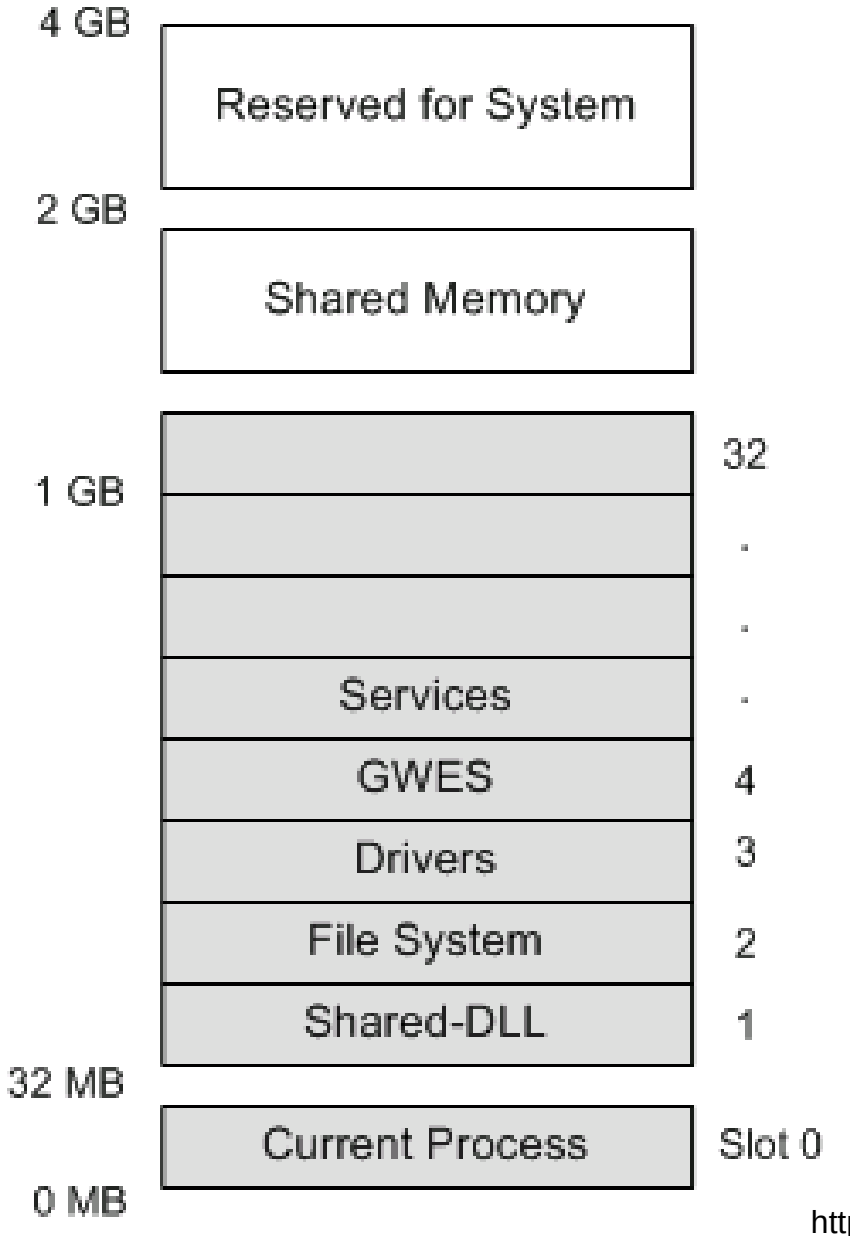
	Windows CE	Windows Mobile
Architektury	ARM, MIPS, x86, SH4	ARM
Modułowość	Tak	Nie
Praca bez GUI	Tak	Nie
Jednolite API	Nie	Tak
Licencja	Dystrybutor	Microsoft



Architektura Windows Mobile

http://www.windowsfordevices.com/sponsors/SP7571057230-AT7810851813.html

Windows CE 5 – organizacja pamięci



- 32 MB na proces (1 GB w CE 6)
- Do 32 procesów (32k w CE 6)
- System *slotów*

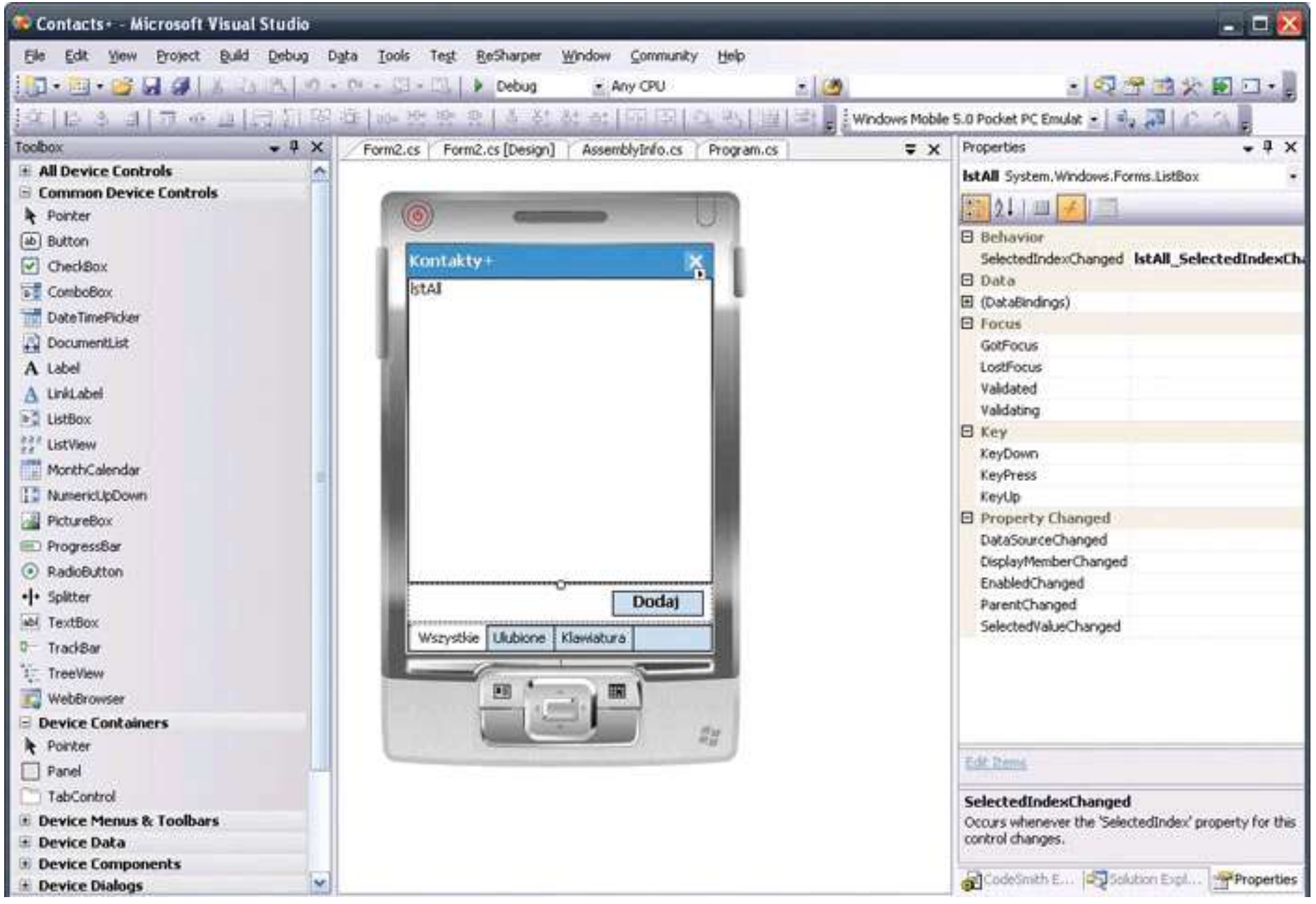
http://www.windowsfordevices.com/sponsors/SP7571057230-AT7810851813.html

Scheduler w Windows CE

- Szeregowanie wątków
- 256 poziomów priorytetu (mniejsze numery ważniejsze)
- Priorytet procesu nie ma znaczenia (inaczej niż w NT)
- *Round-robin*
- *Boosting* i *decay* – zmiana priorytetu w czasie wykonania przez Balance Set Manager
  (ale mniej intensywnie niż w NT)

http://www.sriramkrishnan.com/blog/2006/08/tale-of-two-schedulers-win_115489794858863433.html

Pisanie aplikacji na Windows Mobile



Windows Mobile

Zalety

- Integruje się z innymi produktami MS
- Dobre IDE (Visual Studio)

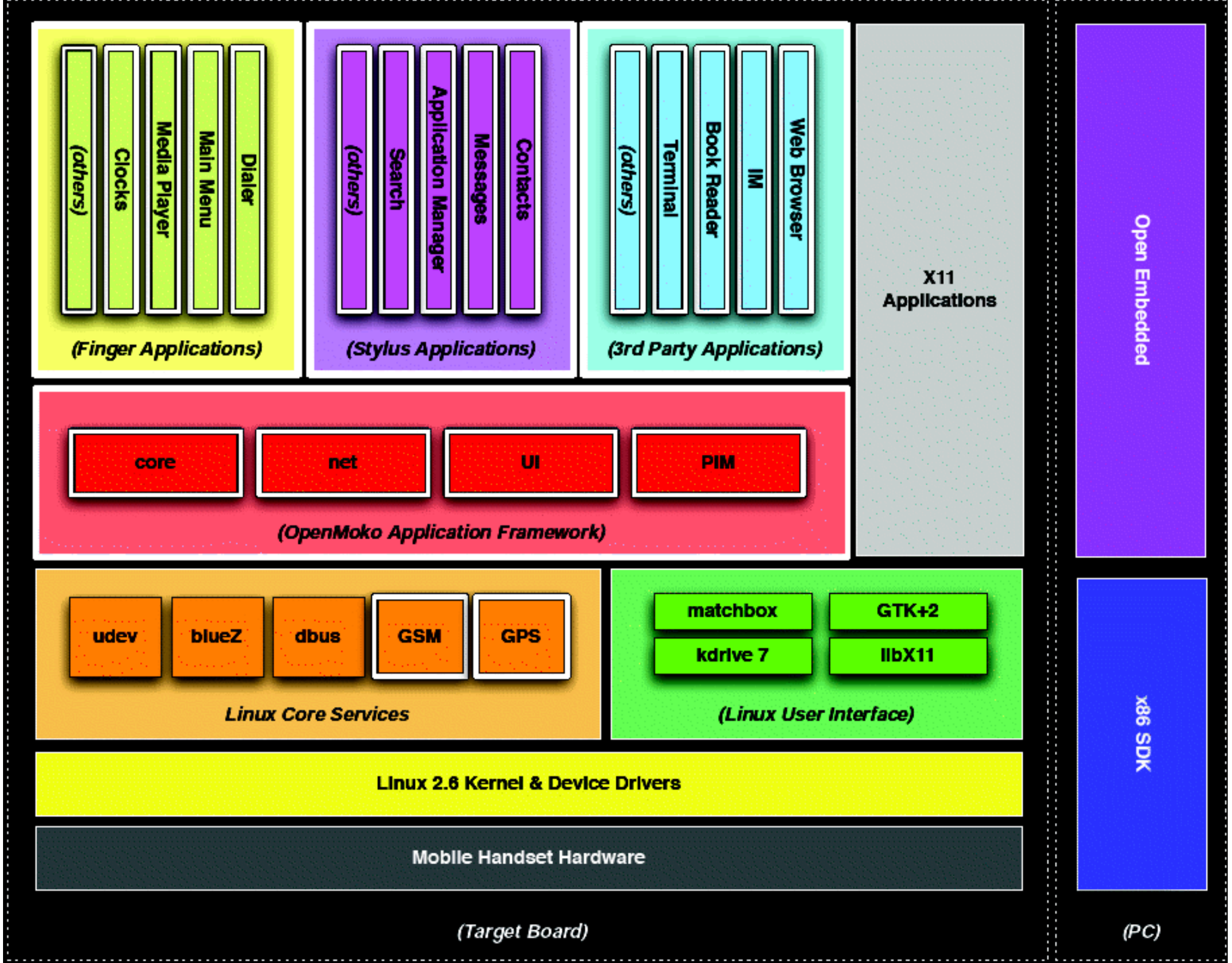
Wady

- Zamknięte źródła
- Brak wsparcia dla technologii firm trzecich
- Programowanie w praktyce wymaga podpisania NDA (dokumentacja)

openmoko

„Pierwszy otwarty telefon”

- Smartfon zaprojektowany przez OpenMoko Inc.
- Udostępniona dokumentacja sprzętu
- Produkowany przez firmę FIC
- Szereg działających dystrybucji (Om2008.12, QtExtended, Debian, ...)

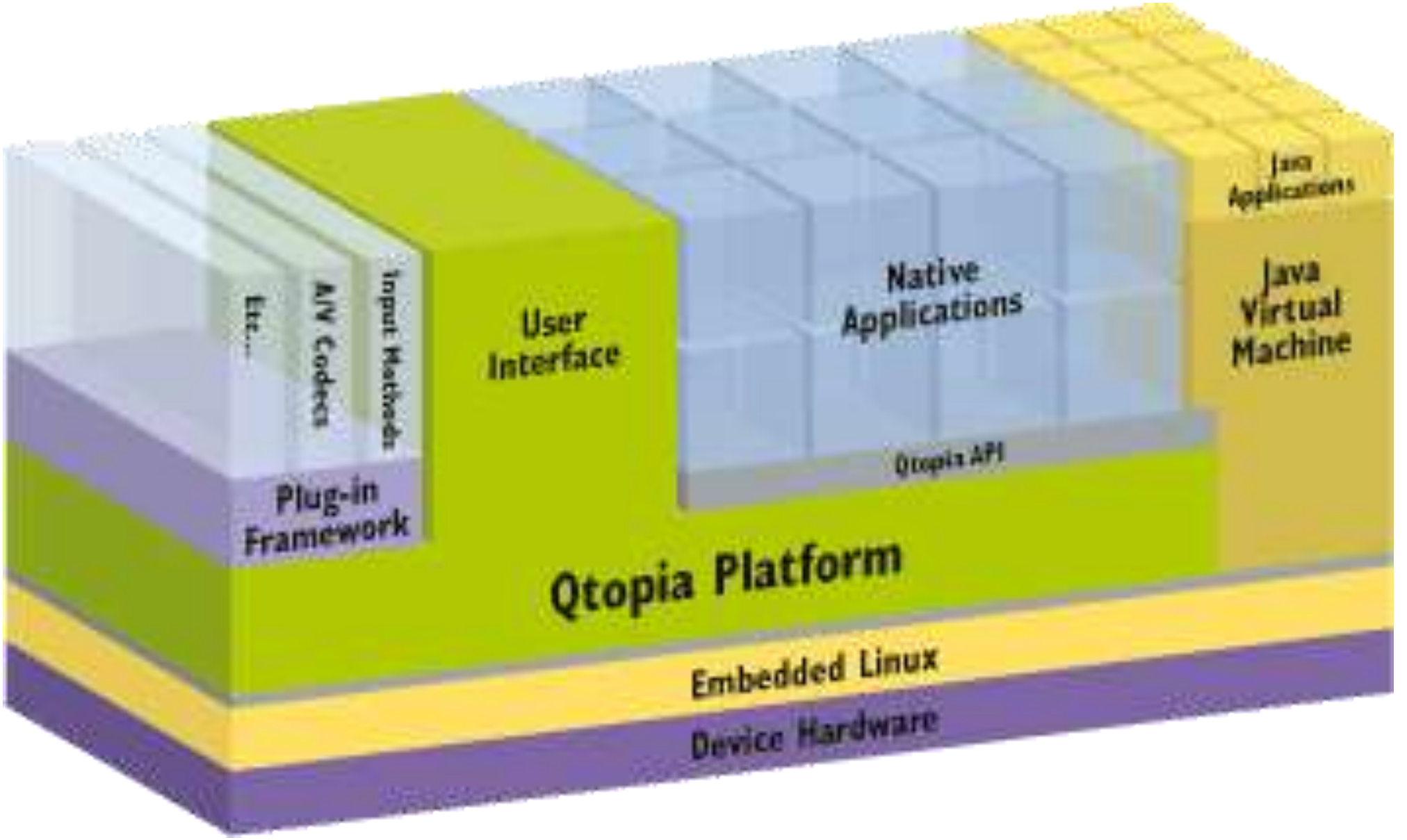

(others)
Clocks
Media Player
Main Menu
Dialer
(Finger Applications)
(others)
Search
Application Manager
Messages
Contacts
(Stylus Applications)
(others)
Terminal
Book Reader
IM
Web Browser
(3rd Party Applications)
X11 Applications
core
net
UI
PIM
(OpenMoko Application Framework)
udev
blueZ
dbus
GSM
GPS
Linux Core Services
matchbox
GTK+2
kdrive 7
libX11
(Linux User Interface)
Linux 2.6 Kernel & Device Drivers
Mobile Handset Hardware
(Target Board)
Open Embedded
x86 SDK
(PC)

OpenMoko Linux

- Kernel: vanilla 2.6.21.3 + kilka łatek ze sterownikami urządzeń
- Standardowy Xorg
- Mini-manager pakietów opkg
- Framework GSM zapożyczony z QtExtended

```
General setup
 →Configure standard kernel features
  (for small systems)
   →...
```

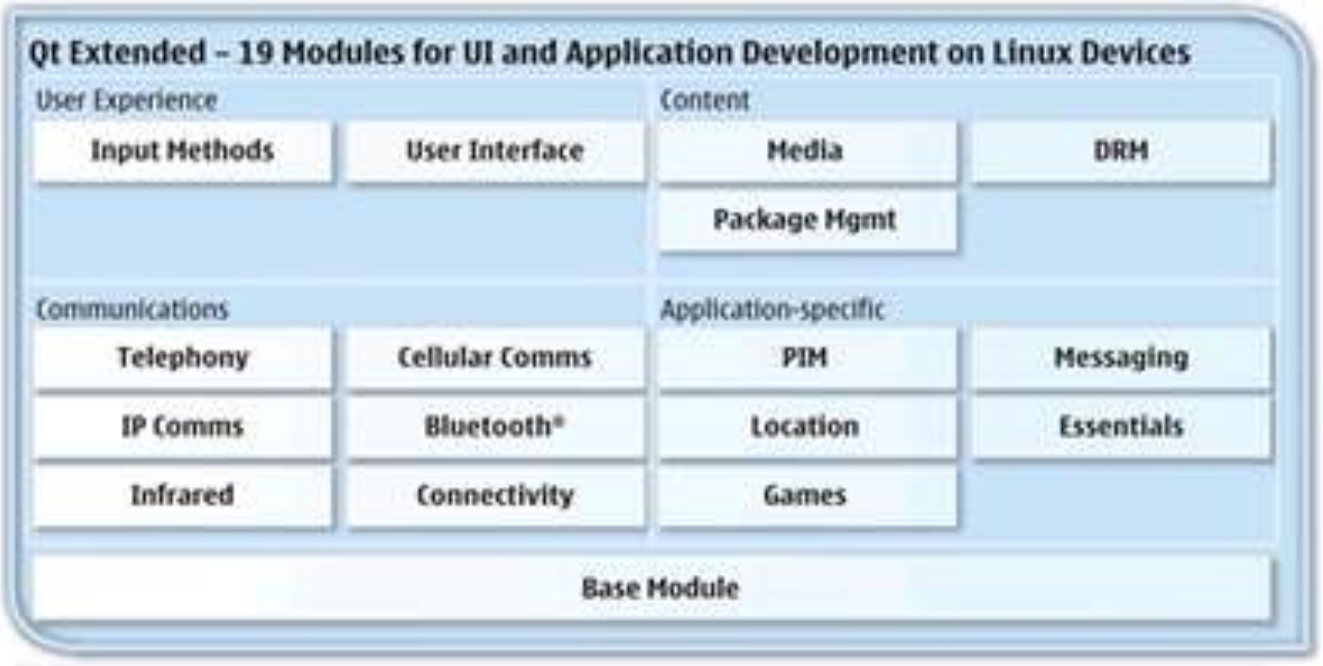
QtExtended (dawniej Qtopia)



HelloWorld w QtExtended

```
#include <QApplication>
#include <QPushButton>

int main(int argc, char **argv)
{
    QApplication a(argc, argv);
    QPushButton hello("Hello world!", 0);
    hello.resize(100, 30);
    a.setMainWidget(&hello);
    hello.show();
    return a.exec();
}
```



OpenMoko Linux

Zalety

- Bazuje na linuksie

Wady

- Bazuje na linuksie

OpenMoko Linux

Zalety

- Bazuje na linuksie
- Łatwość programowania
- Przenośność oprogramowania
- Duża dowolność konfiguracji
- Fajna zabawa

Wady

- Bazuje na linuksie
- Problemy ze sprzętem (oszczędzanie energii, niezawodność GSM, ...)
- Niezbyt ergonomiczny

SYMBIAN OS

Historia

- SIBO
- MC laptop, Series 3, Series 3c, Sienna, Series 3mx
- zalety: zarządzanie mocą, współpraca z innymi komputerami i urządzeniami
- programowanie oparte na C i programowaniu obiektowym
- EPOC
- wsparcie dla urządzeń z ekranem dotykowym
- multimedialny
- zaprojektowany jako obiektowy
- napisany w C++
- bardziej przenośny
- Symbian OS
- Psion + Nokia, Ericsson, Motorola, Panasonic
- przejęcie systemu EPOC i jego dalszy rozwój

Wiadomości ogólne

- przeznaczony na urządzenia smartphone
- system czasu rzeczywistego (real-time system)
- jądro czasu rzeczywistego od wersji
- „Symbian OS v9 onwards”
- wielozadaniowy, wielowątkowy system operacyjny
-
- wykorzystanie mikrojądra
- serwery aplikacji uruchamiane poza mikrojądrem
-
- zorientowany obiektowo
- projekt głęboko przeniknięty ideą abstrakcji

Wiadomości ogólne - c.d.

- brak implementacji pamięci wirtualnej
- wspiera użycie maszyn wirtualnych
- „computer within computer”
-
- wywołania funkcji systemowych
- 1) executive call
- 2) żądanie kernel-service
-
- implementacja capabilities
- (od wersji 9.1)
-
- multimedialność

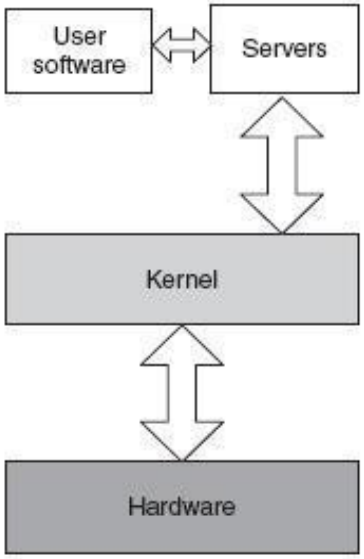


Figure 2.2 Structure of a microkernel

Jądro Symbian OS

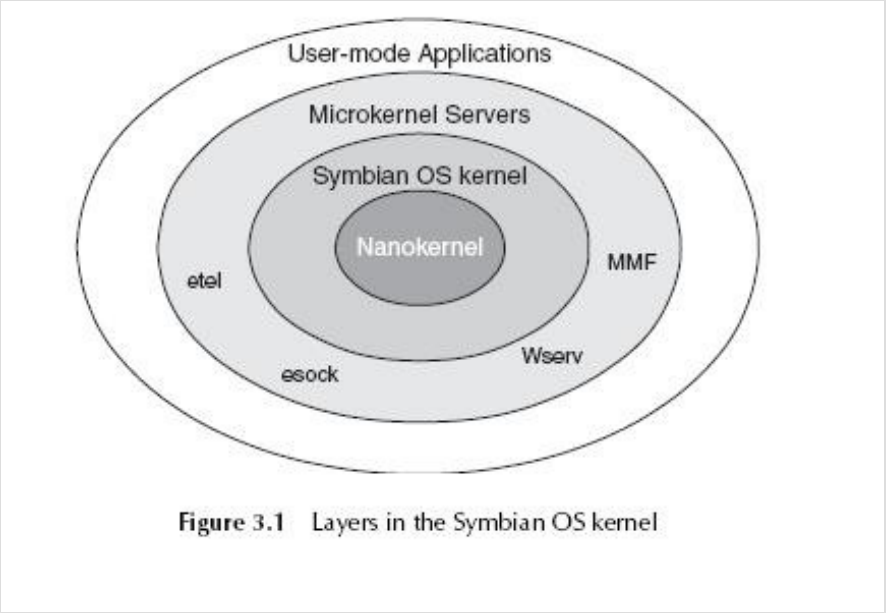


Figure 3.1 Layers in the Symbian OS kernel

-
- nanojądro
- warstwa Symbian OS kernel
- warstwa serwerów
- aplikacje użytkownika
-
-
- aktywne komponenty jądra
- pasywne komponenty jądra
-

Struktura jądra

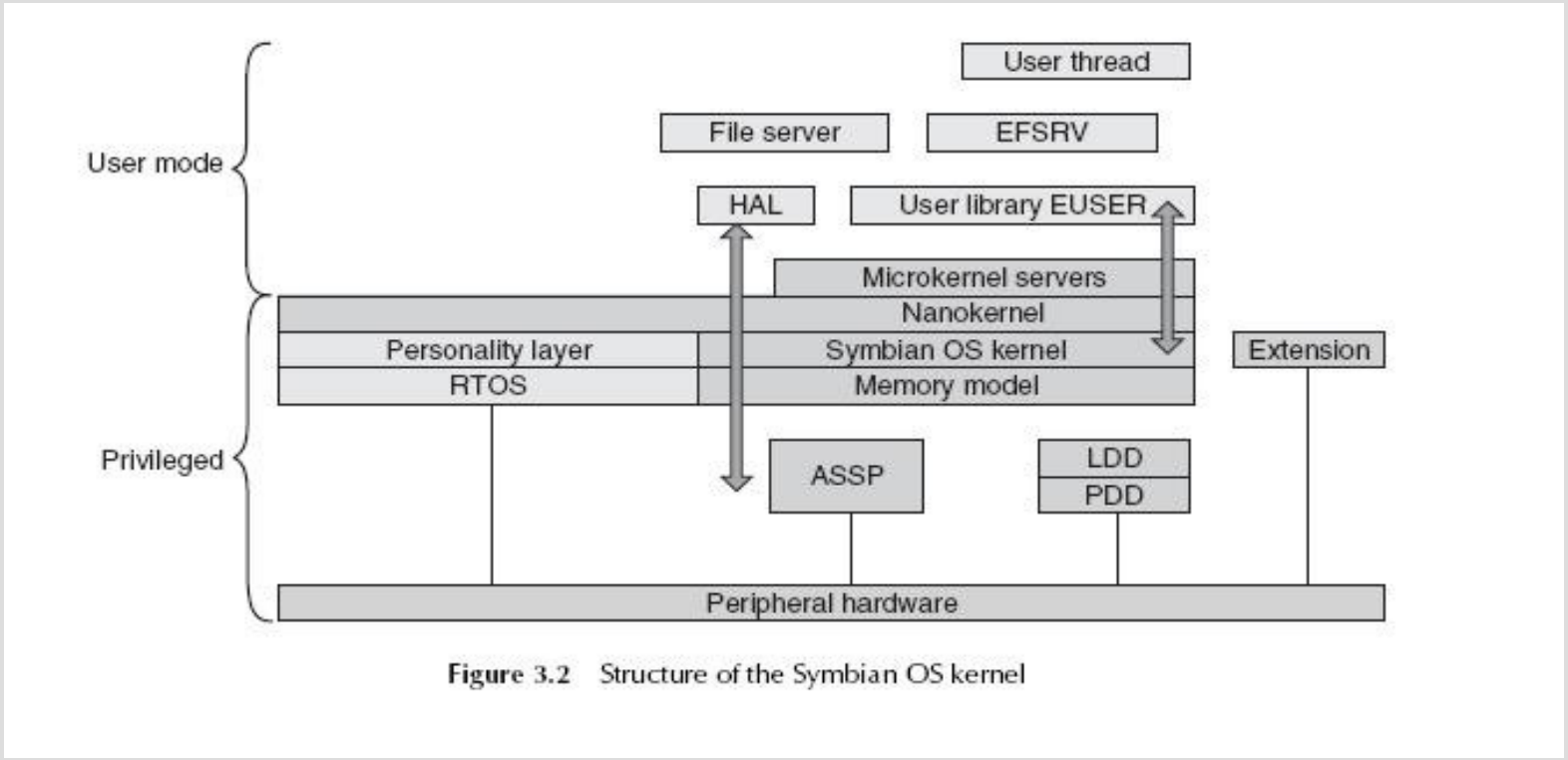


Figure 3.2 Structure of the Symbian OS kernel

- sterowniki urządzeń
- application-specific standard product
- real-time components

Wątki i procesy

- główny wątek i jego pochodne
-
- active objects – wątki lekkie
- zachowanie „aktywnych obiektów”
- współpraca ze schedulerami
-
- nanowątki – ultralekkie procesy
- wątki nabudowane na nanowątkach
- procesy i koncepcje forkowania i łączenia
-
- active objects a wątki Symbianowe

Scheduling i współbieżność

-
- statyczne, monotoniczne szeregowanie zadań
- przewidywanie czasu działania procesu
- priorytety i wyszukiwanie gotowego procesu
-
-
- wsparcie nanojądra
- oczekiwanie na semaforach i mutexach
- problem synchronizacyjny z tym związany
- „dziedziczenie priorytetów”

Bezpieczeństwo

- brak identyfikacji użytkowników
- „capabilities”
- przestrzeń dla instalatorów oprogramowania
- oczekiwania użytkowników
- „gatekeeper”
- reorganizacja systemu bezpieczeństwa w wersji 9 – mechanizm „signing”
- weryfikacja oprogramowan
- - certyfikacja
- - niezależne testy
- - akceptacja i „podpis”
- „podpisy” a „capabilities”
- „data caging”

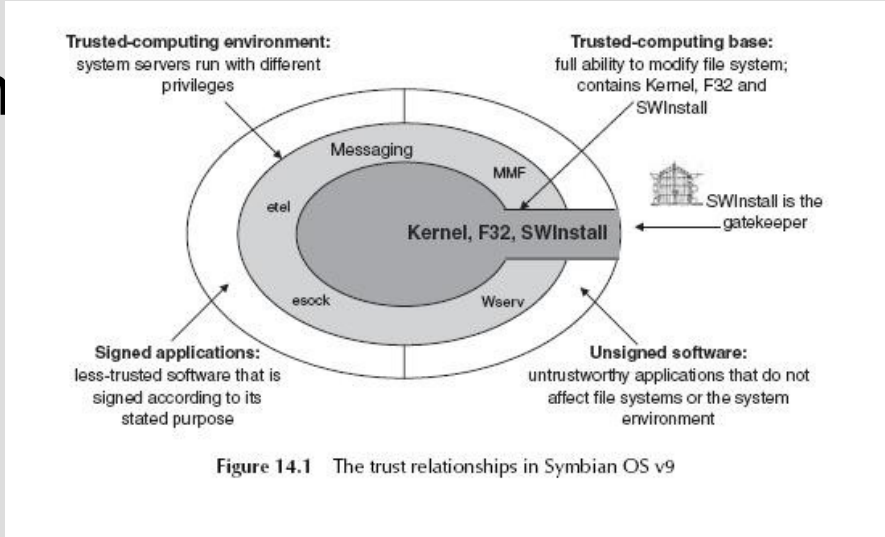


Figure 14.1 The trust relationships in Symbian OS v9

Zarządzanie pamięcią

- brak pamięci wirtualnej
- abstrakcja – adresy
- logiczne i fizyczne
- strategia dwupoziomowego stronicowania
-
- użycie pamięci w aplikacjach
- pamięć dynamiczna
-
- modele pamięci:
- - moving model
- - multiple model

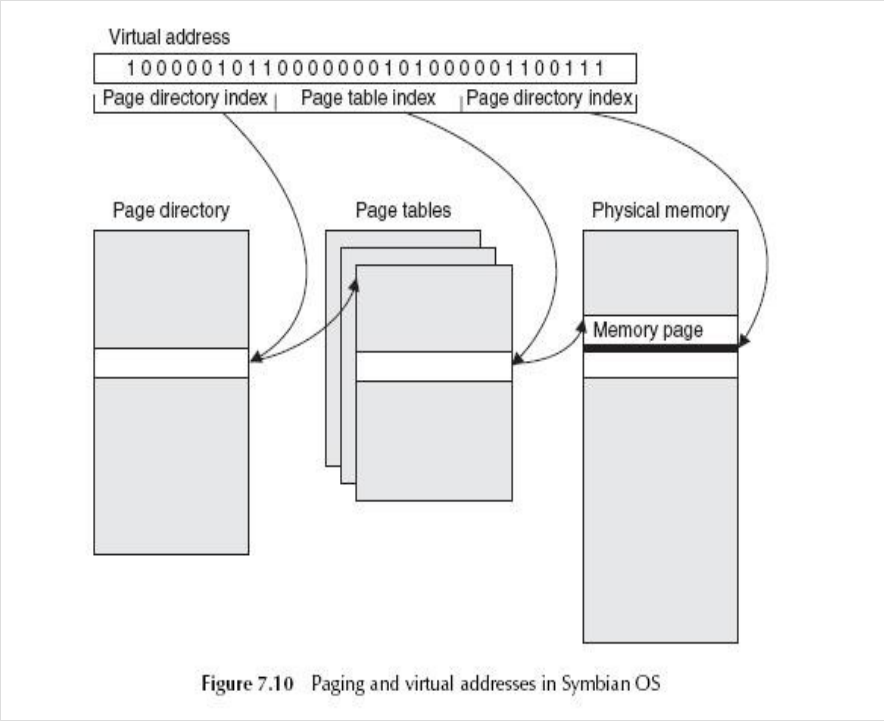


Figure 7.10 Paging and virtual addresses in Symbian OS

I/O w Symbian OS

- sterowniki urządzeń i rozszerzenia jądra
- Harware Abstraction Layer
-
- ruchome nośniki danych
- założenia przyjęte w ich obsłudze:
- - nośnik może być zamontowany
- lub usunięty
- - nośnik może być usunięty
- „na gorąco”
- - każdy nośnik umie zgłosić
- swoje „capabilities”
- - niekompatybilne nośniki musza
- być odrzucone
- - każdy nośnik potrzebuje mocy
- obiekty „socket”

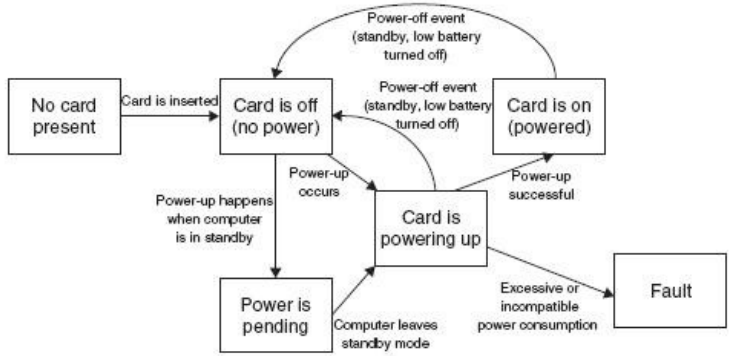


Figure 9.3 Possible power states of removable media on Symbian OS

Systemy plików

-
- kompatybilny z systemem Windows
-
- pamięć flash – sposób działania
- rozwiązanie problemu
-
-
- kompatybilność
- system FAT16
-
- abstrakcja w implementacji systemu plików
- interfejsy typu plug-in

Komunikacja

- Symbian – system do komunikacji
- wsparcie dla: Bluetooth, Ethernet, IR, …
- brak wsparcia dla pamięci dzielonej
- sockety i wsparcie dla protokołów komunikacyjnych
-
- framework do obsługi różnych typów wiadomości
- MTM – message module type
- formaty: email, SMS, fax, BIO,
- interfejs „send-as messaging”
- repozytoria wiadomości
-
- komunikacja międzyprocesowa

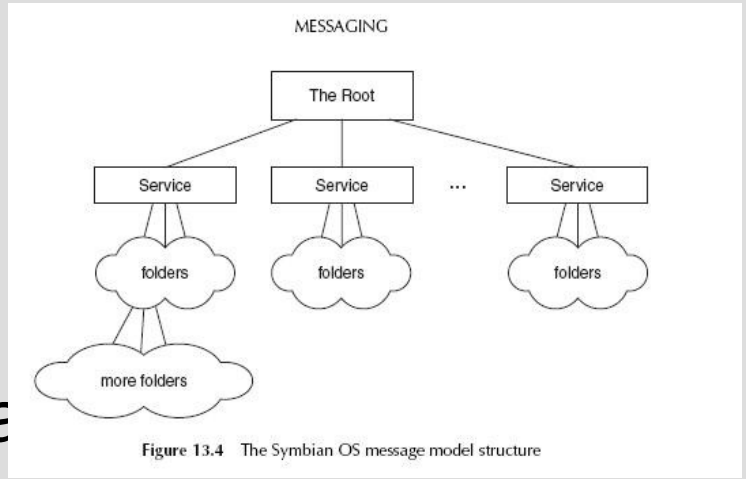


Figure 13.4 The Symbian OS message model structure

Infrastruktura komunikacyjna

- urządzenie
-
- poziom sterowników
-
- implementacja protokołów:
- - moduły CSY
- - moduły TSY
- - moduły PRT
- - moduły MTM
-
- aplikacja użytkownika

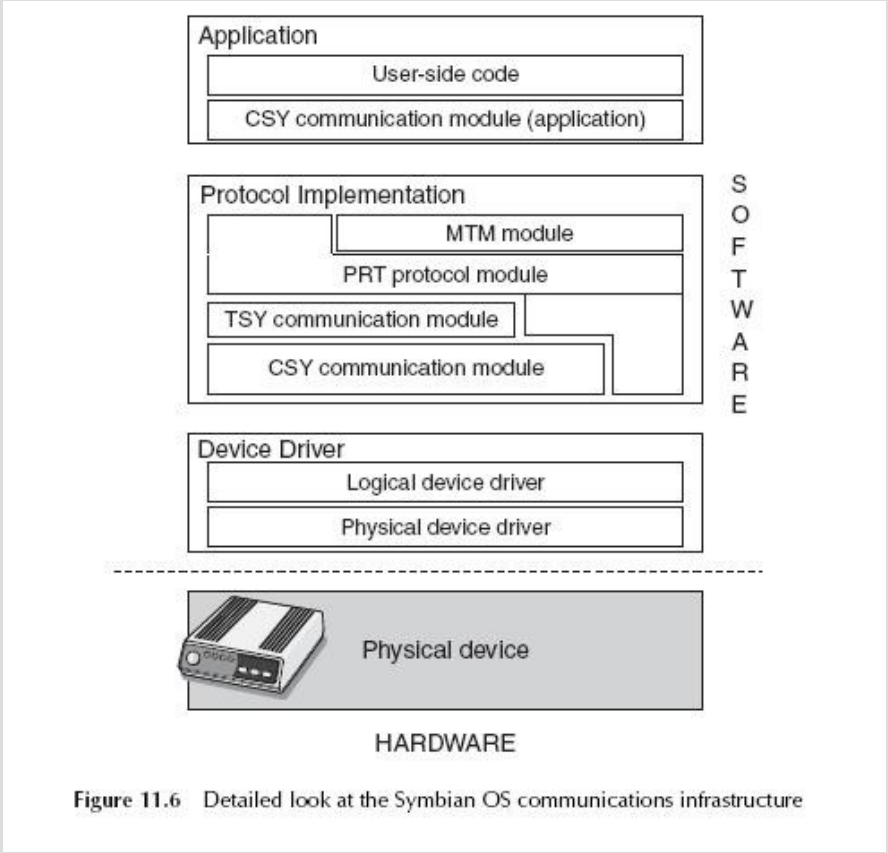


Figure 11.6 Detailed look at the Symbian OS communications infrastructure

Literatura

- M. J. Jipping „Smartphone Operating System Concepts with Symbian OS”


ANDROID

Open Handset Alliance

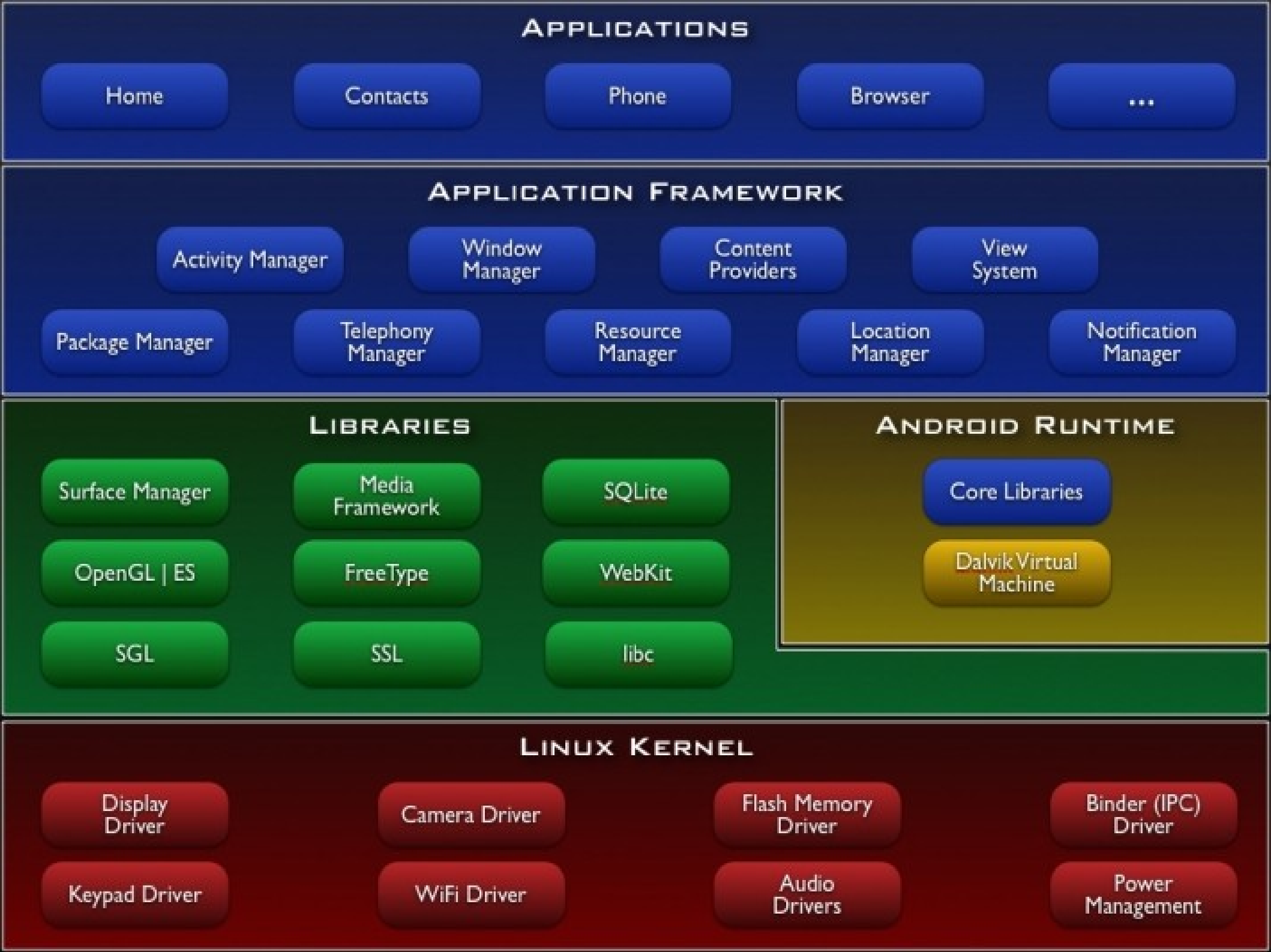
- Google
- Intel
- Motorola
- NVIDIA
- Sony Ericsson
- Motorola
- Qualcomm
- Samsung
- LG Electronics
- Asus
- Toshiba
- T-Mobile
- Vodafone
- ...

Historia

- 05.11.2007 – ogłoszenie prac
- 12.11.2007 – wstępne SDK
- 23.08.2008 – informacje o T-Mobile G1
- 23.08.2008 – Android 1.0 SDK
- 21.10.2008 – uwolnienie kodu
- 22.10.2008 – T-Mobile G1 dostępny
- 05.12.2008 – Android Dev Phone 1
-

Źródła Androida

- Publiczne repozytorium (Git)
-
- Dla każdego modułu osoba nadzorująca
-
- Dokładne zasady co do stylu programowania


Applications
Home
Contacts
Phone
Browser
...
Application Framework
Activity Manager
Window Manager
Content Providers
View System
Package Manager
Telephony Manager
Resource Manager
Location Manager
Notification Manager
Libraries
Surface Manager
Media Framework
SQLite
OpenGL | ES
FreeType
WebKit
SGL
SSL
libc
Android Runtime
Core Libraries
Dalvik Virtual Machine
Linux Kernel
Display Driver
Camera Driver
Flash Memory Driver
Binder (IPC) Driver
Keypad Driver
WiFi Driver
Audio Drivers
Power Management

Jądro linuksa

- Wersja 2.6
- zarządzanie pamięcią
- zarządzanie procesami
- abstrakcja sprzętu

Biblioteki

- Napisane w C/C++
-
- **libc**
  - zoptymailzowane pod urządzenia wbudowane
-
- **media**
  - nagrywanie i odtwarzanie wielu formatów
-
- **grafika 3D**
  - na bazie OpenGL-a
-
- **SQLite**
  - silnik relacyjnej bazy danych

Dalvik VM

-
- Uruchamia aplikacje Javy skonwertowane do formatu *Dalvik Executable (.dex)*
-
- Przystosowana do pracy w systemach z małą ilością pamięci i słabym procesorem.
  -
- Narzędzie **dx**
  - .class -> .dex
  - może być wiele klas w jednym dexie
  - usuwanie zduplikowanego kodu
-
  -

Application Framework

- Komponenty GUI
-
- Wymiana danych z innymi aplikacjami
-
- Zarządzanie zasobami
-

Aplikacje

- Piszemy w Javie
-
- Korzystamy z Application Framework
-
- Na równi z dostarczonymi aplikacjami
-
- Każda aplikacja to
  - oddzielny wątek
  - własna instancja VM
  - plik .apk

Bezpieczeństwo

- Tylko jawnie udostępniona funkcjonalność aplikacji jest dostępna dla innych
-
- Wymagane podpisywanie aplikacji
-
- Domyślnie aplikacji nie wolno prawie nic

Bezpieczeństwo

-
- Każda aplikacja ma wsłasne user id w linuksie przydzielane w momencie instalacji
-
- Wszystkie pliki stworzone przez aplikację też mają to samo user id.
-
- Chyba, ze użyjemy atrybutu `sharedUserId`
-

Bezpieczeństwo

- Elementy permission i uses-permission w AndroidManifest.xml
-
- Użytkownik jest pytany czy udzielić dostępu do danej usługi
-
- URI Permissions

Licencja

- Apache v2 open source license
  - commercial-friendly
  - brak klauzuli copyleft
  - nie jest wirusogenna

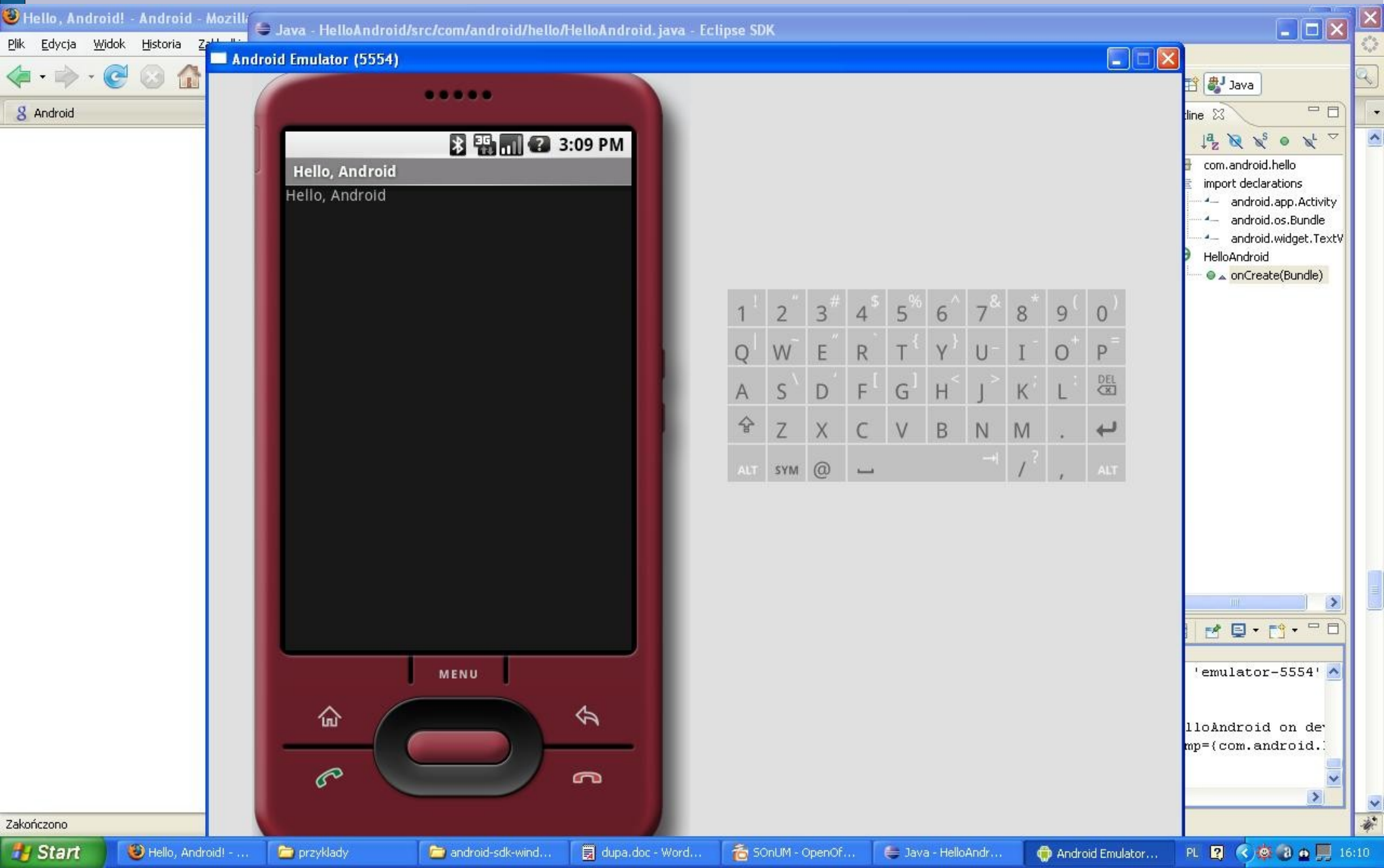
Hello Android

```
package com.android.hello;

import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class HelloAndroid extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        TextView tv = new TextView(this);
        tv.setText("Hello, Android");
        setContentView(tv);
    }
}
```

Wynik


Hello, Android! - Android - Mozilla
Java - HelloAndroid/src/com/android/hello/HelloAndroid.java - Eclipse SDK
Android Emulator (5554)
3:09 PM
Hello, Android
Hello, Android
MENU
com.android.hello
import declarations
android.app.Activity
android.os.Bundle
HelloAndroid
onCreate(Bundle)
Zakończono
Start

KONIEC

Systemy operacyjne na urządzenia mobilne na przykładzie Symbian OS - na podstawie książki „Smartphone Operating System Concepts with Symbian OS” M. J. Jipping

Wojciech Tyczyński

13 stycznia 2009

1 Historia

1. SIBO
    - ujrzał światło dzienne w roku 1988
    - stworzony przez Psion Computers
    - uruchamiany na małych urządzeniach
    - pierwszy komputer na którym wykorzystano SIBO to MC laptop, ale komputer okazał się porażką
    - następnie system był wykorzystywany kolejno na komputerach:
        - Series 3 - rok 1991, mały komputer mieszczący się w kieszeni (half-VGA-sized monitor)
        - Series 3c - rok 1996, dodatkowo wprowadzono obsługę podczerwieni
        - Sienna - rok 1996, mniejszy ekran, bardziej przypominał organizator niż komputer
        - Series 3mx - rok 1998, zdecydowanie szybszy procesor
    - dobrze zarządzał mocą
    - dobrze współpracował z innymi komputerami i urządzeniami podręcznymi
    - programowanie oparte na C oraz idei programowania obiektowego
2. EPOC
    - system 32-bitowy
    - wspierał urządzenia z ekranem dotykowym
    - multimedialny
    - znacznie bardziej przenośny, działał na dużo większej ilości platform
    - zaprojektowany jako obiektowy, napisany w C++
3. Symbian OS
    - przedsięwzięcie Symbian
    - Psion wraz z liderami rynku telefonii komórkowej: Nokia, Ericsson, Motorola, Panasonic
    - przejęcie systemu EPOC i jego dalszy rozwój już pod nazwą „Symbian OS”
    - skierowany na kilka ogólnych platform sprzętowych
    - wystarczająco elastyczny by zaspokoić wymagania przemysłu
    - jednocześnie udostępnia producentom możliwość odróżnienia swoich produktów od produktów innych producentów

2 Symbian OS - wiadomości ogólne

- system operacyjny z założenia przeznaczony na urządzenia smartphone
- urządzenia smartphone - łączą w sobie funkcje telefonu komórkowego, poczty elektronicznej, przeglądarki sieciowej, pagera, GPS, aparatu fotograficznego, kamery wideo, ...
- system operacyjny czasu rzeczywistego (read-time system)
- nałożone są bardzo surowe wymagania na czas wykonywania poszczególnych operacji
- jeżeli funkcje albo urządzenia nie spełniają wymogów nie mogą być w ogóle wykorzystane
- na początku Symbian OS nie był systemem czasy rzeczywistego, ale od wersji „Symbian OS v9 onwards” już wykorzystuje on jądra czasu rzeczywistego

- wielozadaniowy, wielowątkowy system operacyjny
- wiele procesów może być uruchomionych współbieżnie
- system operacyjny oparty na mikrojądrze, które dostarcza tylko ograniczony zbiór funkcji systemowych
- pozostała funkcjonalność (która jest dostępna w jadrach monolitycznych) jest zapewniana przez serwery aplikacji uruchamiane poza mikrojadrem
- w Symbian OS, kiedy użytkownik wywołuje funkcję „open" aby uzyskać dostęp do portu, najpierw musi połączyć się z serwerem, który zapewnia do niego dostęp, dopiero potem może otworzyć port
- takie podejście wprowadza dodatkowe ułatwienie: aby zmienić np. sposób komunikacji pomiędzy urządzeniami, trzeba zmienić tylko kod odpowiedniego serwera i go zrekompilować - nie musimy wprowadzać zmian w mikrojądrze
- ponadto podejście to w dużym stopniu wspiera kocepcje abstrakcji i modularności
- system zorientowany obiektowo (cecha odziedziczona z EPOC)
- idea abstrakcji bardzo głęboko przenika w projekt systemu - wywołania funkcji systemowych używają obiektów systemu lub jądra
- tak jak Unixie w momencie otwierania pliku jest tworzony jego deskryptor i jest on używany jako parametr funkcji open(), tak w Symbianie utworzony zostanie obiekt klasy RFile i na nim zostanie wywołana metoda open()
- z powodu ograniczonej przestrzeni dyskowej, nie ma implementacji pamięci wirtualnej
- mimo to Symbian OS wspiera wykorzystanie maszyn wirtualnych - implementacja „computer within a computer"
- implementacja Javy i jej środowiska run-time potrzebnego do uruchomienia Javy jest właśnie zrealizowana za pomocą tego machanizmu
- Symbian OS wspiera dwa rodzaje JVM: PersonalJava i Java Platform Micro Edition (Java ME); maszyna wirtualna Javy jest interesująca ze względu na to, że nie posiada wewnątrz systemu operacyjnego; jest tylko abstrakcyjnym komputerem, który zapewnia warstwę pośrednią pomiędzy programem w Javie a platformą sprzętową i systemem operacyjnym
- istnieją dwa rodzaje wywołań funkcji systemowych:
  - executive call - wysyła zadanie do jądra, aby to wykonało ono uprzywilejowane operacje na rzecz procesu z przestrzeni użytkownika, który wywołał funkcję; jej wywołanie powoduje przerwanie softwarowe, które jest następnie obsługiwane, po czym sterowanie jest przekazywane z powrotem do procesu; wywołania te mogą modyfikować obiekty z przestrzeni jądra, natomiast nie mogą ich ani usuwać ani też tworzyć
  - żądanie kernel-service - jest to np. alokacja pamięci albo utworzenie wątku; prośba o realizacje tego żądania jest kierowana do specjalnego serwera, który zajmuje się zarządzaniem zasobami systemowymi oraz ich ochroną przed niepowołanym dostępem
- Symbian OS od wersji 9.1 implementuje „capabilities", o których więcej w sekcji „Bezpieczeństwo"
- każdemu żądaniu musi towarzyszyć odpowiednia zdolność, aby mogło być ono zrealizowane
- zapewnia to, że podczas wykonywania programu użytkownika nie może on skłonić jądra do zrobienia wszystkiego czego sobie zapragnie, część zadań jądra jest chronionych;
- Symbian OS został zaprojektowany jako rdzeń z API dla multimediów

- urządzenia multimedialne są zarządzane przez specjalne serwery i framework, który pozwala użytkownikom na implementację modułów opisujacych nowe i istniejace juz zasoby multimedialne oraz sposoby ich wykorzystania i przetwarzania

3 Struktura jądra

- główną składową jądra jest nanojądro, które zapewnia najbardziej podstawowe funkcje
- w skład tych funkcji wchodzą przede wszystkim podstawowe operacje synchronizacyjne: mutexy i semafory, operacje dotyczące planowania, obsługa sygnałów
- funkcje tu są tak prymitywne, że nie potrzebujemy tutaj żadnych skomplikowanych operacji takich jak np. dynamiczna alokacja pamięci
- na nanojądrze nabudowana jest właściwa warstwa mikrojądra - „Symbian OS kernel layer"; funkcje z tej warstwy wykorzystują funkcje pochodzące z nanojądra do stworzenia bardziej skomplikowanych, w tym przede wszystkim: obsługę wątków użytkownika, scheduling procesów, zmianę kontekstu, komunikację międzyprocesową, pamięć dynamiczną, dynamicznie ładowane biblioteki
- warto zauważyć, że przerwania mogą zmusić tę warstwę do przeplanowania czynności do wykonania i może się to zdarzyć w każdej chwili, w szczególności podczas przełączania kontekstu
- warstwa serwerów - używa operacji jądra sporadycznie, dlatego może być wydzielona; funkcje z tej warstwy pokrywają pewne z góry określone pola funkcjonalności, np. pracę z portami i najczęściej są uruchamian jako usługi dla aplikacji użytkownika
- ostatnią warstwą są właśnie wspomniane powyżej aplkacje użytkownika

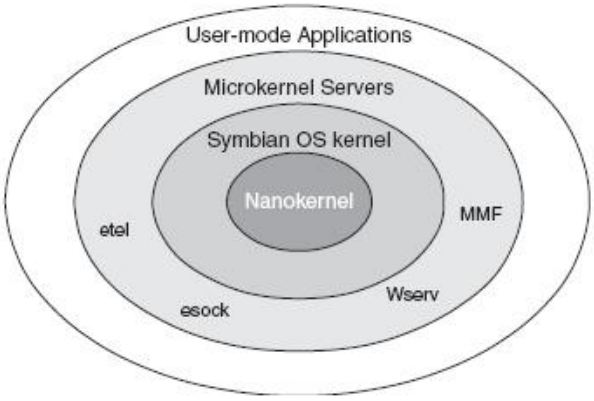


Figure 3.1 Layers in the Symbian OS kernel

- trzeba wspomnieć, że tylko niewielkie części jądra chodzą bez przerwy; większa icg część jest zaimplementowana pasywnie - wykonuje przypisane im operacje po odpowiednich wywołaniach systemowych lub przerwaniach
- aktywne komponenty jądra, czyli te które chodzą bez przerwy, odpowiadają przede wszystkim za komunikację międzyprocesową, zarządzanie pamięcią globalną, przełączanie kontekstu a także wielowątkość
- wielowątkowość - dzięki niej jest możliwe np. jednoczesne pobieranie danych z karty pamięci i odbieranie połączenia telefonicznego, w przeciwnym przypadku trzeba by priorytetować operacje i wykonywać tylko jedną z nich co byłoby dość uciążliwe dla użytkowników

- pasywne komponenty jądra - nie wykonują się bez przerwy, ale są dostępne do wykonania w momencie nadejścia odpowiedniego żądania
- są wywoływane przez wywołania systemowe lub przerwania
- do pasywnych komponentów należą przede wszystkim serwery mikrojądra, które uruchamiane są tylko wtedy, gdy są potrzebne, a po wykonaniu przeznaczonego dla nich zadania są wyłączane
- podobnie jest ze sterownikami urządzeń, które są ładowane dynamicznie w momencie, kiedy zaczynami używać urządzenie; w Symbianie mamy dwa rodzaje sterowników: logiczne (implementujące operacje abstrakcyjne) i fizyczne (implementujących operacje logiczne już dla konkretnych urządzeń), ale o tym później
- struktura jądra Symbian OS:
  - sterowniki urządzeń: PDD i LDD
  - ASSP - application-specific standard product - dzięki temu elementowi jądro może się bezpośredonio komunikować ze sprzętem
  - real-time componenet - komponenty związane z obsługą rozmów telefonicznych, mogą się również komunikować bezpośrednio ze sprzętem, ale tylko jeżeli zostaną uruchomione w specjalnym trybie (personality layer)
  - RTOS i personality layer służą właśnie implementacji operacji związanych z funkcjonalnością telefonu; RTOS implementuje funkcje GSM w bezpośrednim powiązaniu ze sprzętem

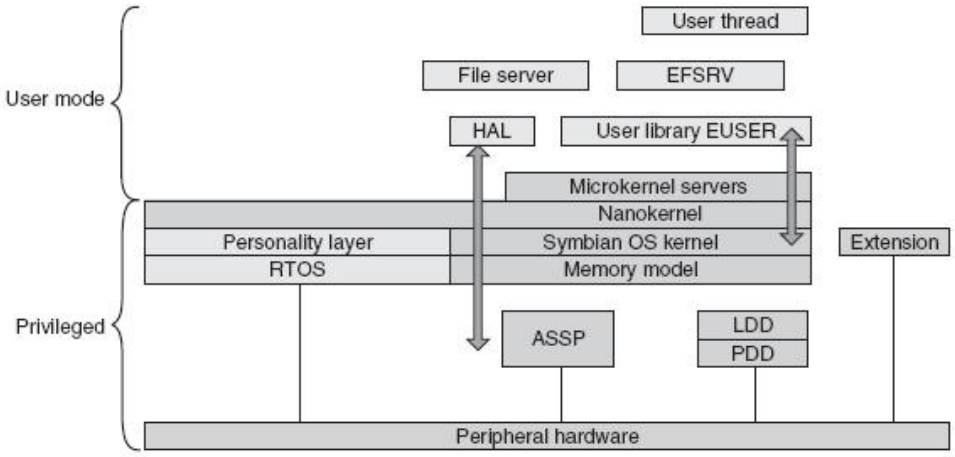


Figure 3.2 Structure of the Symbian OS kernel

4 Wątki i procesy

- w systemach, gdzie musimy się bardziej liczyć z zasobami duże znaczenie ma "waga"wątków i procesów
- z reguły istnieje jeden główny wątek, który kontroluje większość aspektów aplikacji; jeżeli wątków jest więcej są one zazwyczaj jego pochodnymi
- w Symbian OS istnieje specjalny rodzaj wątków - lekkie wątki tzw. „active objects”

- każdy „active object” jest skierowany na zdarzenie, które może powodować, że wątek lub proces zostanie zablokowany (komunikacja, operacje IO, ...); podczas oczekiwania na zajście tych zdarzeń wszystkie obiekty aktywne wewnątrz danego procesu dla systemu zachowuja się jak jeden wątek
- każdy „active object” pracuje z pojedynczym aktywnym schedulerem, który nasłuchuje zdarzeń, na które czeka dany wątek; wszystkie active objects w obrębie jednego procesu używaja tego samego schedulera, ale dla różnych procesów mogą to być różne schedulery
- każdy obiekt aktywny definiuje punkt wejścia w kodzie, który jest używany przez schedulera, jeżeli zostanie wygenerowane oczekiwane zdarzenie
- nanojądro zapewnia bardzo podstawowe funkcje dla nanowątkow: scheduling, synchronizację i timing serwis
- nanowątki są uruchamiane w trybie uprzywilejowanym
- możliwe stany nanowątków: suspended, fast Semaphore Wait, DFS Wait (oczekiwanie na delayed function call), sleeping i inne
- nanowątek jest ultralekkim procesem, posiada mini-kontekst
- każdy wątek Symbian OS jest nabudowany na nanowątku - dodaje do stosu nanowątku swój stos wywołań
- wątki mogą operować w przestrzeni jądra za pomocą wywolań systemowych
- co ciekawe, dodają siedem nowych stanów, aby zwrócić uwagę na rodzaje blokad jakim mogą podlegać wątki
- procesy to wątki zgrupowane pod jedną strukturą PCB (process control block) i z jedną przestrzenią pamięci
- w procesie jest wyróżniony jeden wątek jako punkt startowy procesu
- wątki współdzielą parametry schedulingu jak również obiekty przestrzeni pamięci
- kiedy proces się kończy, jądro ubija wszystkie wątki danego procesu
- Symbian OS wspiera koncepcje forkowania i łączenia procesow, z tym że istnieje znaczna różnica: przy forkowaniu struktura PCB nie jest duplikowana - tworzony jest nowy świeży wykonywalny obraz
- „active objects” są w rzeczywistosci wątkami Symbian OS, ale istnieją korzyści z ich wykorzystywania: kiedy oczekują one na zdarzenie system operacyjny nie musi sprawdzać czy są one już gotowe, a zajmuje się nimi gdy dane zdarzenie już rzeczywiście zajdzie - mniej thread checkingu
- idea korzystania z nich polega na tym, że muszą one się same zrzekać sterowania na rzecz systemu, kiedy już są gotowe na oczekiwanie na określone zdarzenie

5 Scheduling i współbieżność

- Symbian OS jest systemem operacyjnym czasu rzeczywistego
- kombinacja funkcjonalności ogólnego przeznaczenia z systemem czasu rzeczywistego sprawia, że najlepszym wyborem jest system który używa „statycznej i monotonicznej strategii szeregowania zadań” - organizuje procesy z najkrótszym deadlinem najpierw
- to działa jeżeli system może przewidzieć, jak długo będzie działał dany proces; przewidywanie to bazuje oczywiście na samym procesie, ale też charaktyerystyce systemu - przeiwydwalne muszą być:

1. czas oczekiwania - czas od przerwania do przekazania sterowania do wątku użytkownika oraz czas od przerwania do przekazania sterowania do wątku jądra
2. czas na uzyskanie informacji o procesie - np. czas potrzebny na znalezienie gotowego wątku o najwyższym priorytecie
3. czas przenoszenia wątkow pomiędzy kolejkami i procesorem

- mechanizm szeregowania zadań wprowadza kwanty czasu - procesy z tym samym deadlinem (albo bez niego) mogą mieć przypisane kwanty czasu i podlegać priorytetowemu szeregowaniu
- w Symbian OS są 64 poziomy priorytetów
- wyszukiwanie gotowego procesu odbywa się na podstawie maski bitowej, która zawiera informację czy w danej kolejce (która istnieje dla każdej wartości priorytetu) znajduje się jakikolwiek gotowy do wykonania proces
- z powodu wsparcia nanojądra istnieje wiele rodzajów semaforów na obu poziomach jadra
- te najbardziej prymitywne obiekty są w nanojądrze: mutexy i semafory
- nanojądro wspiera zarówno blokujące jak i nieblokujące operacje na mutexach i semaforach
- w Symbian OS problemem jest fakt, że jeżeli proces o niższym priorytecie posiada mutexa, może on opóźniać proces oczekujacy na jego zwolnienie o wyższym priorytecie; ten problem został rozwiązany tak, że Symbian OS używa twz. „dziedziczenia priorytetów” - polega to na tym, że w momencie, kiedy proces o niskim priorytecie otrzymuje mutexa, system operacyjny zwiększa mu priorytet do najwyższej wartości priorytetu spośród procesów oczekujących na ten mutex - to ma zapewniać że żaden inny proces nie zostanie uruchomiony zanim nasz proces nie zwolni mutexa

6 Bezpieczeństwo

- ponieważ telefony są w założeniu przeznaczone dla jednego użytkownika, w Symbian OS nie ma zaimplementowanej identyfikacji użytkowników; brak tej identyfikacji oznacza ze jest tylko jeden zestaw uprawnień - ten sam dla wszystkich
- zamiast identyfikacji Symbian OS, od wersji 9, nadaje aplikacji w momencie instalacji określone „capabilities”; zdolności są sprawdzane przy każdym wywołaniu systemowym; jeżeli program wywołuje funkcję systemową, do której nie posiada uprawnień, aplikacja skutkuje błędem
- ponadto istnieje przestrzeń, do której dobranie się wymaga specjalnej zdolności, któraąposiadają tylko aplikacje instalujące software w systemie; dzięki temu niesystemowe programy (np. wirusy) nie mogą zainfekowac zainstalowanych już aplikacji
- Symbian OS zakłada identyfikacje na podstawie funkcji procesow i na tej podstawie umożliwia dostęp do zasobów
- istnieją procesy użytkownika, procesy systemowe i inne, które mają więcej uprawnień niż procesy użytkownika, ale mniej niż systemowe
- użytkownicy oczekują możliwości użycia wszystkich funkcjonalności bez uwierzytelniania; ale systemy te są narażone na działanie różnego typu wirusów; dlatego musimy sobie jakoś z tym radzić
- w wersjach wcześniejszych od 9, Symbian OS oferowal „gatekeeper’a” - pytał o pozwolenie dla każdej instalowanej aplikacji a zatem zakładano, że tylko zainstalowane aplikacje moga siać spustoszenie w systemie
- mimo licznych zalet takiego podejścia problemem jest fakt, ze użytkownik nie zawsze wie czy dana aplikacja jest bezpieczna - mimo iż posiada ona użyteczne funkcjonalności może "po cichu"dokonywać spustoszenia w systemie

- w wersji 9 platforma bezpieczeństwa została całkowicie zmieniona; mimo że „gatekeeper” nadal istnieje, to odpowiedzialność za weryfikację instalowanego oprogramowania została zdjęta z użytkownika - programiści są teraz zobligowani do weryfikacji ich softwaru poprzez tzw. mechnizm „signing” i system weryfikuje tylko stwierdzenia deweloperów; taka weryfikacja jest jednak wymagana tylko od oprogramowania korzystającego z określonych funkcji systemowych
- proces wyreyfikacji programowania:
  1. programista musi uzyskać ID sprzedawcy od istytucji wydajacej certyfikaty (Symbian)
  2. po stworzeniu oprogramowania musi wysłać je do niezależnych testów - wysyła ID sprzedawcy, software i listę sposobów w jakie software odwołuje się do systemu
  3. podczas testów lista ta jest weryfikowana - jeżeli weryfikacja jest możliwa i zakończona sukcesem software zostaje „podpisany” i odesłany sprzedawcy jako gotowy do dystrybucji
- w momencie instalacji sprawdzany jest „podpis” i jeżeli jest on ważny, pakietowi zostają nadane odpowiednie „capabilities”
- warto zauważyc, że jest kilka poziomów uwierzytelniania - np. aplikacje, które nie odwołują się do zasobów systemowych (np. tylko wyświetlają coś na ekran) nie wymagają takich „podpisow”

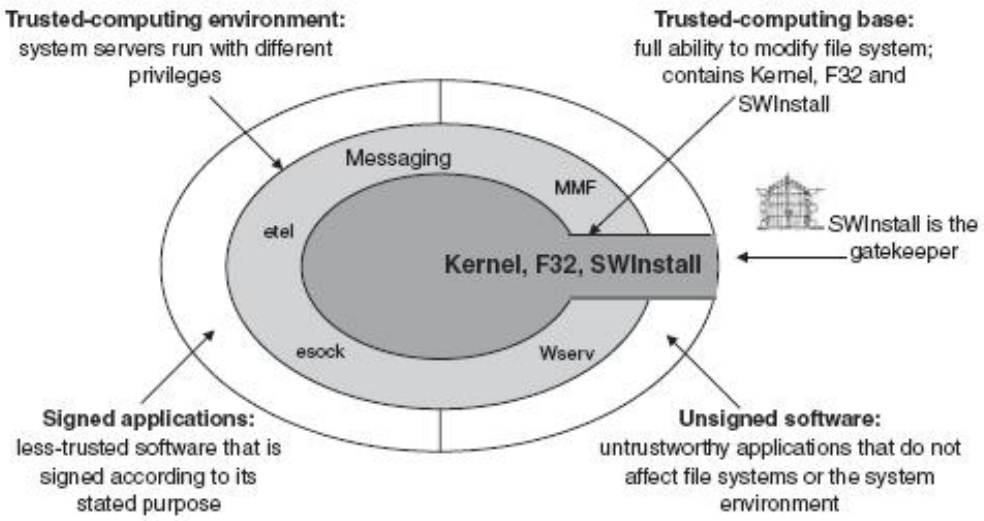


Figure 14.1 The trust relationships in Symbian OS v9

- dodatkowym zabezpieczeniem jest twz. „data caging” - organizacja danych w „directories”; wykonujące się kody istnieją tylko w jednym określonym katalogu, ponadto dane zapisywane przez aplikacje mogą być zapisane też tylko w jednym katalogu, do którego dostępu nie mają żadne inne programy

7 Zarządzanie pamięcią

- Symbian OS jest idealnym przykładem systemu, który nie używa mechanizmu wirtualnej pamięci
- używa natomiast tej samej abstrakcji co większe systemy - programy posługują się adresami logicznymi, które system operacyjny mapuje na adresy fizyczne

- programy mogą być umiejscowione w dowolnym miejscu pamięci i nie wiedzą one, w którym dokładnie miejscu się znajdują, dlatego użycie adresów logicznych jest niezbędne
- pamięć jest dzielona na strony logiczne i ramki fizyczne - ramki przewaznie zajmują 4kB, ale nie jest to reguła
- strategia dwupoziomowego stronicowania:
  - pierwszy poziom - 12 bitów (index w katalogu stron), katalog stron jest trzymany w pamieci i jest wskazywany przez specjalny rejestr - rejestr translacji tabel (translation table-base register) pozycja w katalogu stron wskazuje na drugi poziom stronicowania - kolekcję stron

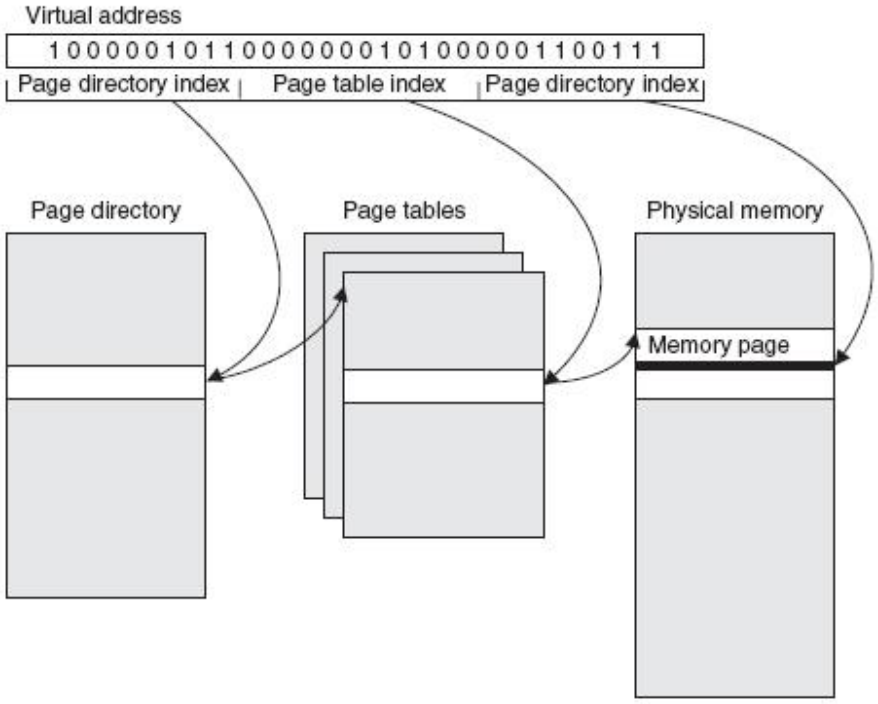


Figure 7.10 Paging and virtual addresses in Symbian OS

- w Symbian OS wszystkie strony pamięci używane przez daną aplikację powinny być załadowane do pamięci w momencie jej uruchomienia; ponieważ Symbian OS może zaadresowac 4GB pamięci, a większość smartphone'ów jeszcze ciągle nie posiada takiej ilości pamięci fizycznej, mogą pojawić się sytuacje, że strony do której się odwołujemy nie ma w pamieci; to skutkuje wyrzuceniem wyjątku, który nie zostaje nigdzie przechwycony i zakończeniem działania aplikacji; jest to KERN-3-EXEC error
- ponieważ rozmiar pamięci dynamicznej przeznaczonej dla programu (np. na stos) jest ograniczony w Sybmianie OS, możliwe jest również używanie statycznych żądan dla pamieci dynamicznej - ramki pamięci są alokowane do stron z listy wolnych ramek (jeżeli nie ma wolnych ramek zgłaszany jest wyjątek)
- istnieją cztery implementacje modelu pamięci używane w Symbian OS - każda przeznaczona dla określonych platform sprzętowych:

- moving model - dla wczesnych architektur ARM (wersja 5 i wczesniejsze); chroniony dostęp do stron przez ustawienie odpowiednich bitów skojarzonych z ramkami pamięci i podział na obszary całego katalogu stron (podział nie byl zrobiony explicite)
- mutliple model - dla późniejszych architektur ARM (6 i późniejsze), MMU się różni od wcześniejszych wersji - katalog stron jest podzielony na dwie osobne części - strony użytkownika i strony jądra
- direct model - zakłada, że nie ma w ogóle MMU (używany podczas developmentu, bo w telefonach brak MMU byłby czymś dziwnym)
- emulator model - wsparcie emulatora Symbian OS na Windowsie; emulator jest uruchamiany jako pojedynczy proces Windowsowy, dlatego przestrzeń adresowa jest ograniczona do 2GB; brak ochrony pamięci - cała jest dostępna dla procesów Symbian OS

8 IO w Symbian OS

- sterowniki urządzeń - wykonują się jak uprzywilejowany kod jądra, dzięki temu mają dostęp do chronionych zasobów systemowych; LDD i PDD; mogą bćc ładowane dynamicznie
- rozszerzenia jądra - sterowniki urządzeń ładowane podczas bootowania; uruchamiane po starcie schedulera; implementują kluczowe funkcjonalności dla systemu - DMA, kontrola szyn do komunikacji z urządzeniami peryferyjnymi (np. USB) itp.
- przyczyny takiego podziału:
  1. oddziela oprogramowanie, które jest zależne od hardwaru umożliwiając uruchamianie odpowiednich sterowników bez rekompilacji całego jądra
  2. uwypukla idee programowania obiektowego
- HAL (Harware Abstraction Layer) - zestaw zmiennych i funkcji służących do dostępu do konfiguracji systemu i jego atrybutów; są grupy przeznaczone specjalnie dla określonych urządzeń sprzętowych (np. klawiatur), są też grupy ogólnego przeznaczenia - dostęp do ogólnych parametrów platformy (np. sterowniki przesyłu informacji, timer)
- DMA - uslugi dostarczane przez sprzęt DMA są wykorzystywane przez PDD, ale pomiędzy nimi są jeszcze dwie warstwy oprogramowania:
  1. warstwa DMA - podzielona na dwie podwarstwy - zależną i niezależną od sprzętu
  2. rozszerzenie jądra - sterownik, który jest uruchamiany jako jeden z pierwszych podaczas bootowania
- nośniki pamięci powodują wiele problemów - do ich obsługi potrzebny jest kontroler, sterownik, szyna, komunikacja poprzez DMA z CPU, ale nośnik może być w każdej chwili usunięty; dlatego system musi jakoś wykrywać moment zamontowania czy usunięcia nośnika; co więcej niektOre sloty moga współpracować z różnymi urządzeniami (np. karty SD, karty miniSD, MultiMediaCard)
- w Symbian OS zostały przyjęte założenia:
  1. wszystkie ruchome nośniki danych mogą być usunięte lub włożone
  2. nośnik może być usunięty na gorąco (w czasie używania)
  3. każdy nośnik umie zglosić swoje „capabilities”
  4. niekompatybilne nośniki muszą być odrzucone
  5. każdy nośnik potrzebuje mocy

- Symbian OS zapewnia kontroler, który komunikuje się ze sterownikami wszystkich nośnikow, również w softwarze; obiekt socket zostaje utworzony w momencie włożenia karty i ten obiekt tworzy kanał przez który będą przepływać potem dane; Symbian OS zapewnia wiele zdarzeń, które pojawiają się w momencie zmiany stanu
- sterowniki są konfigurowane jako active objects

9 System plików

- większość systemów operacyjnych na telefony komórkowe posiada system plikow, który może być dzielony ze środowiskiem Microsoft Windows
- system plikow kompatybilny z Windowsem - FAT32
- pamięć flash nie może po prostu nadpisywać pamięci - musi ją najpierw usunąć, a potem dopiero zapisać; ponadto nie jest np. możliwe usuwanie pojedynczych bajtów - możliwe jest tylko usuwanie całych bloków pamięci a ponadto czas usuwania jest relatywnie duży

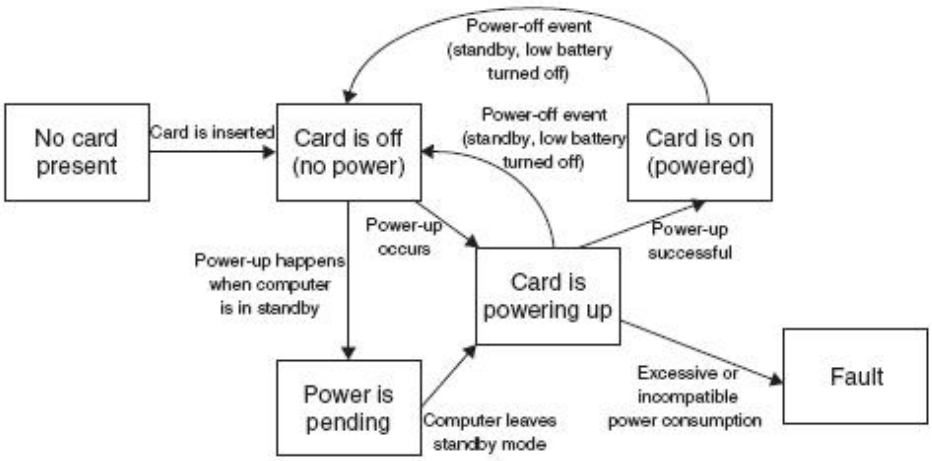


Figure 9.3 Possible power states of removable media on Symbian OS

- aby poradzić sobie z tymi niedogodnościami potrzebny jest specjalny system plików - jego koncepcja polega na tym, że kiedy zawartość pamięci ma zostać zmodyfikowana, system tworzy nową kopię zmienianych danych w nowych blokach pamięci, przemapowuje wskaźniki, natomiast usunięcie starych danych zostawia na chwilę, kiedy będzie miał na to czas
- ponieważ od platform telefonów wymagamy kompatybilnosci z innymi komputerami, najczęściej używany jest w nich system FAT
- Symbian jako system operacyjny na platformy telefonów musi wspierać system FAT16 i rzeczywiście wspiera; w rzeczywistości używa tego systemu plików do przechowywania większości swoich danych;
- warto zauważyć, że serwer plikow w Symbian OS jest zaimplementowany w sposób bardzo abstrakcyjny - pozwala to na używanie różnych implementacji systemu plików, które mogą nawet współistnieć wewnątrz tego samego serweru plików

- Symbian OS wspiera też inne implementacje systemu plikow za pomocą interfejsów typu plug-in; w szczególności istnieją implementacje NFS i SMB dla Symbiana

10 Komunikacja i jej infrastruktura

- Symbian OS to system operacyjny, którego głównym zadaniem jest właśnie komunikacja
- Symbian OS wspiera wiele rodzajów komunikacji
- jako system operacyjny dla telefonów wpiera oczywiście telefonię, ale również Bluetooth, łączność bezprzewodową, Ethernet, podczerwien i wiele innych protokołów; co więcej pozwala implementować nowe technologie, które będą używane w przyszłości
- Symbian OS nie wspiera wykorzytywania pamięci dzielonej - system operacyjny dla telefonów z założenia zostaż zaprojektowany do pracy w izolacji
- Symbian OS posiada też bogaty zestaw implementacji dla sieciowych operacji I/O - jego podstawowe struktury komunikacyjne - gniazda (sockets) - zostały zaimplementowane dla wszystkich możliwych rodzajów komunikacji, wspierajac tym samym wiele protokołów komunikacyjnych, w szczegolnosci TCP/IP, Bluetooth, WAP, HTTP, ...
- Symbian OS posiada bardzo ogólny framework, który zawiera komponenty osbługujące wiele różnych typów wiadomości
- obiektowość wymusza, że każdy typ wiadomości zawiera własna implementację tych ogólnych komponentów
- MTM - message type module - definiowanie typów wiadomości
  1. CBaseMtmUi - interfejs użytkownika i definicje „capabilities”
  2. CBaseMtm - interfejs kliencki pomiędzy wewnetrzną reprezentacją wiadomości a interfejsem użytkownika
  3. CBaseMtmUiData - dostęp do wlaściwości danych z interfejsu użytkownika (np. ikon)
  4. CBaseServerMtm - transport wiadomości po stronie serwera
- Symbian OS wspiera cztery główne formaty - email, SMS, fax i BIO jak również wiele innych
- ponieważ w systemie jest mnóstwo implementacji MTM, a musimy być w stanie szybko wyszukać aktualnie nam potrzebne, Symbian OS zapewnia specjalny rejestr zainstalowanych MTM'ów dostępnych dla aplikacji
- ponadto istnieje interfejs śend-as messaging- prosty interfejs który pozwala aplikacjom na stworzenie wiadomości do wysłania przy użyciu prostego API; programista informuje tylko system operacyjny o typie tej wiadomości, natomiast system sam wybiera odpowiednią implementację i wysyła wiadomość
- repozytoria wiadomości to zasób dzielony przez procesy - dlatego musi istnieć serwer, który zapewnia do nich dostęp i jednocześnie chroni to repozytorium:
  - dostęp do repozytorium danych wiadomości - serwer przydziela dostęp do danej wiadomości na wyłączność
  - dostęp do MTM'ów - umożliwia aplikacjom identyfikację żądań (np. wysylanie wiadomości), do których potrzebujemy funkcjonalności odpowiednich protokołów i odpowiedniego MTM'u
- komunikacja asynchroniczna

- wiadomości są przechowywane w folderach, do których dostęp jest tak jakby były dziećmi serwisów, na które można patrzeć jak na źródła wiadomości
- pluggowlna architektura wiadomości - można dodawać nowe rodzaje wiadomości poprzez zaimplementowanie nowego modułu dynamicznie ładowanego przez serwer wiadomości (messaging server)
- komunikacja miedzyprocesowa - centralny problem dla implementacji Symbiana, który oparty jest na mikrojądrze, ponieważ wiele funkcjonalności jest zepchniętych na serwery uruchamiane na poziomie użytkownika, które muszą się ze sobą komunikować
- obiekty z przestrzeni użytkownika używają systemu socket-based
- serwery przestrzeni użytkownika są „active objects", które oczekują na żądania i je realizują
- dane są wymieniane poprzez gniazda przez obiekty z klasy RMessage2
- Łączność wewnątrz Symbian OS

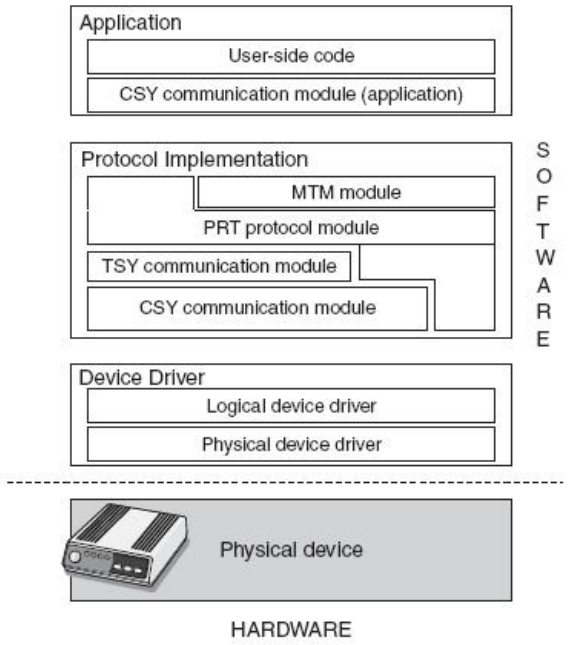


Figure 11.6 Detailed look at the Symbian OS communications infrastructure

- na dole jest urządzenie, np. modem telefonu, transmiter Bluetooth
- dalej jest poziom sterownika urządzenia - software pracujący bezpośrednio ze sprzętem poprzez ustandaryzowany interfejs; Symbian zapewnia sterowniki do najbardziej popularnych części hardwaru; dzięki standaryzacji jeden sterownik może obsługiwać wiele urządzeń
- podział na PDD i LDD (LDD mogą być takie same dla pewnych klas urządzeń - np. urządzenia serializowalne - port IR i port RS232)
- następna warstwa - implementacja protokołu:
  1. moduły CSY - moduły niskiego poziomu - komunikują się za pomocą sterowników bezpośrenio ze sprzętem
  2. moduły TSY - implementują funkcje związane z telefonią, np. inicjowanie czy kończenie rozmów

3. moduły PRT - implementacja protokołów
4. MTM - moduły implementujace mechanizm obsługi wiadomości - od czytanie do powiadamiania o postępie w ich wysyłaniu; odpowiadają za adresowanie, tworzenie, odpowiadanie na wiadomosci jak i obsługę folderów

- aplikacja korzystająca z infrastruktury komunikacyjnej

11 Słownik

- ARM - 32-bitowy procesor RISC Acorn RISC Machine
- MMU - Memory Managment Unit
- BIO - wiadomości, które są przeznaczone dla sprzętu, a nie dla użytkownika