Systemy operacyjne na urządzenia mobilne

Alicja Łuszczak
Juliusz Sompolski
Piotr Świgoń

14 stycznia 2009

Urządzenia mobilne

- Niewielkich rozmiarów, z ograniczonymi możliwościami względem komputerów.
- Zazwyczaj z procesorami o architekturze RISC, obecnie głównie ARM.
- Interakcja z użytkownikiem zapewniona zwykle przez ekrany dotykowe, miniaturowe klawiatury, pady wielokierunkowe.
- Możliwość komunikacji.

Wymagania

- Łatwy i intuicyjny interfejs.
- Wbudowane oprogramowanie o funkcjonalności organizera – kalendarze, notatki, kontakty etc.
- Możliwość przechowywania danych, np. na kartach pamięci.
- Rozbudowane możliwości łączności z innymi urządzeniami – kabel szeregowy, USB, IrDA, Bluetooth, WiFi.

Wymagania

- Możliwość instalacji zewnętrznych aplikacji.
- Obecnie zazwyczaj również funkcje telefonu komórkowego.

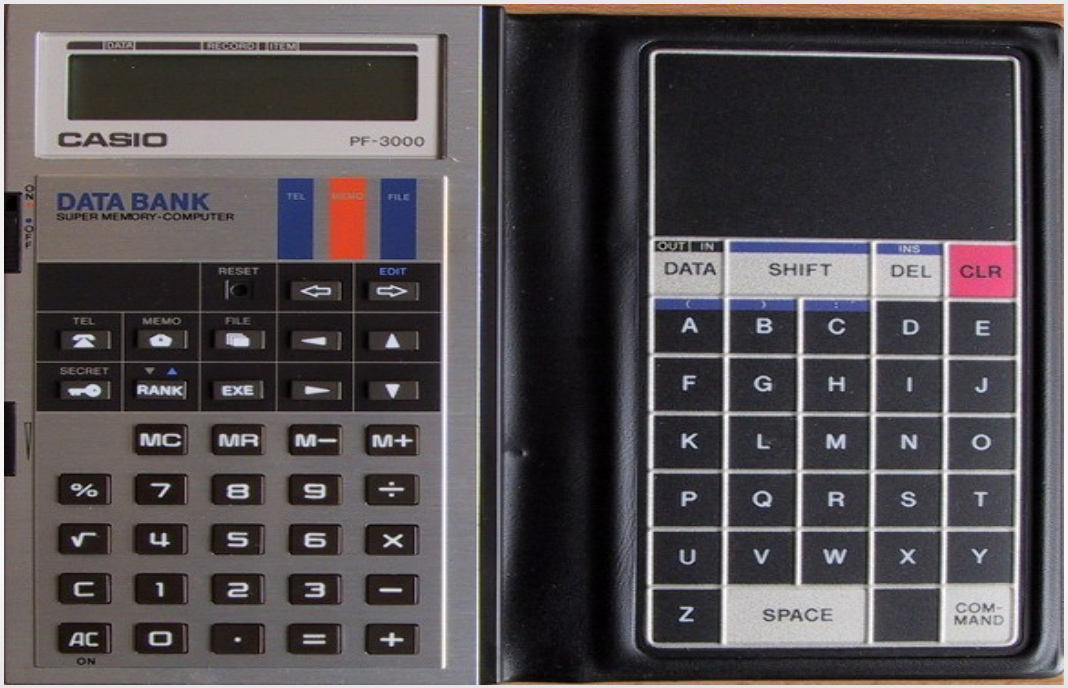
- Wszystko to powinno działać niezawodnie przy ograniczonych możliwościach sprzętowych.

Historia

- Za pierwsze PDA uważany jest Casio PF-3000 z 1983 roku.
- Terminy PDA i palmtop ukute zostały w 1992 roku, wraz z Apple Newton – Newton OS.
- W 1992 roku IBM zaprezentował też pierwszy *smartfon* o nazwie Simon, z systemem Zaurus OS.
- W 1996 Nokia wypuściła swój pierwszy telefon z serii Communicator – Nokia Communicator 9000 na systemie GEOSTM 3.0.

Historia

Apple Newton oraz IBM Simon



Pierwsze PDA – Casio PF-3000

Historia

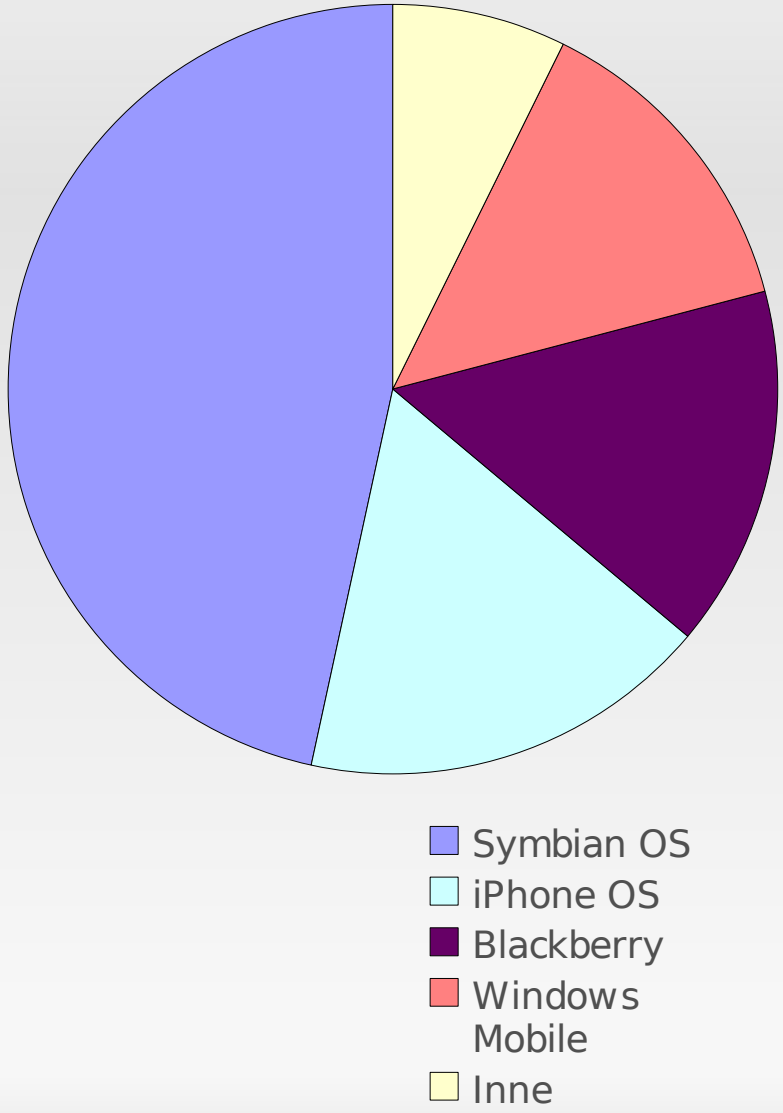
Nokia 9000 Communicator z 1996 roku

Oraz dla porównania najnowszy
Nokia Communicator
– Nokia Communicator E-90 z 2007 roku

Systemy operacyjne

Dane sprzedaży
za III kwartał 2008:

- Symbian OS – 46.6%
- iPhone OS – 17.3 %
- Blackberry – 15.2 %
- Windows Mobile – 13.6 %



Smartfony

Nokia E71 z Symbianem z S60

Sony Ericsson P800 z Symbianem z UIQ

Smartfony

iPhone z iPhone OS

T-Mobile G1 z Androidem

Mark Lennihan / AP

Smartfony

BlackBerry 8800 z RIM BlackBerry

Samsung Omnia z Windowsem Mobile

PDA

PalmOne Tungsten T5 i Palm OS

HP iPAQ 214 z Windowsem Mobile

Przegląd wybranych systemów


ANDROID

Open Handset Alliance

- Porozumienie 47 firm dążących do stworzenia otwartych standardów dla telefonii mobilnej.

- Motorola
- Samsung
- Sony Ericsson
- Toshiba
- LG
- ASUSTek
- Google
- Nvidia
- Broadcom
- Synaptics
- Vodafone
- T-Mobile

Sprzęt – T-Mobile G1

- Dostępny w sprzedaży od październ-ernika 2008.
- W tej chwili jedyny telefon z Androidem.

Sprzęt – co można podhakować

Na niektórych urządzeniach można zainstalować Androida „na dziko”.

- Motorola A1200 Ming
- HTC Vogue
- Nokia N810
- Asus EEEPC 1000H
- I inne

Sprzęt – przyszłość

Wielu producentów zadeklarowało wypuszczenie na rynek telefonów opartych na Androidzie.

- Kogan (Agora i Agora Pro – 29.I.2009)
- Motorola
- Lenovo
- Sony Ericsson (lato 2009)
- Samsung (drugi kwartał 2009)
- HTC (lato 2009)
- I inni

Android SDK

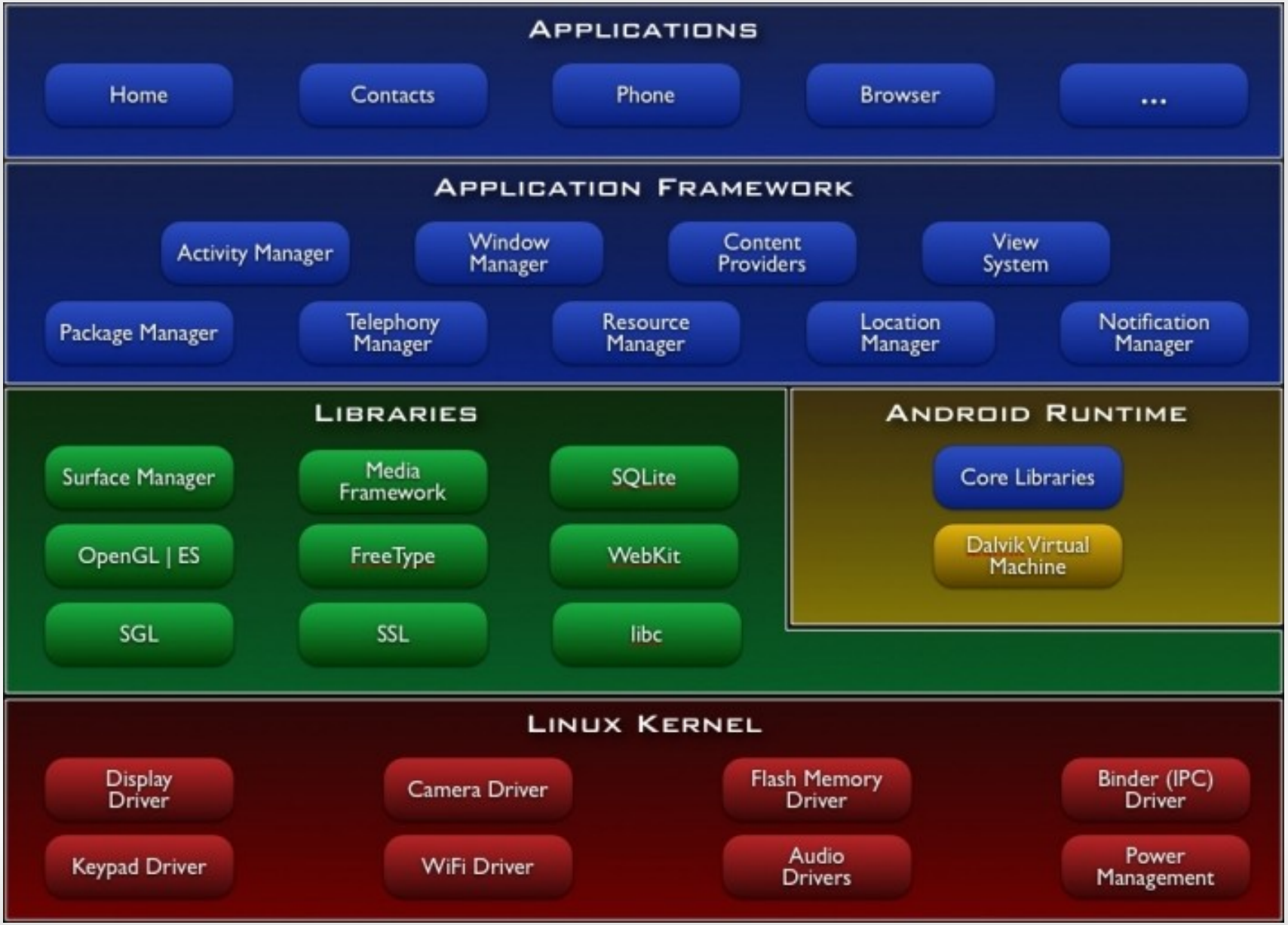
W skład Android Software Development Kit wchodzą:

- Debugger
- Biblioteki
- Emulator telefonu oparty na QEMU
- Dokumentacja
- Przykładowy kod
- Tutoriale

Android SDK

- Przeznaczony do pracy na komputerach opartych na procesorach x86 z systemem operacyjnym Linux, Mac OS 10.4.8 lub nowszym, Windows XP lub Vista.
- Oficjalnie wspierane jest środowisko Eclipse (wersja 3.2 lub nowsza) z zainstalowaną wtyczką Android Development Tools.

Architektura systemu



Biblioteki

Android zawiera zbiór bibliotek napisanych w C i C++, z których twórcy aplikacji mogą korzystać za pośrednictwem frameworka aplikacji.

- Biblioteka systemowa C – oparta na libc z BSD.
- Biblioteki multimedialne – oparte na OpenCORE. Umożliwiają obsługę m.in. formatów MPEG4, H.264, MP3, AAC, AMR, JPG i PNG.
- SQLite – relacyjna baza danych dostępna dla wszystkich aplikacji.

Biblioteki

- Surface Manager – zarządza dostępem do wyświetlacza i łączy warstwy graficzne różnych aplikacji.
- LibWebCore – nowoczesny silnik przeglądarki internetowej.
- Scalable Graphics Library – biblioteka grafiki 2D.
- Biblioteki 3D – oparte o OpenGL ES 1.0. Mogą wykorzystywać sprzętową akceleracje 3D.
- FreeType – do renderowania czcionek.

Framework aplikacji: Views

- Komponenty do budowy interfejsu użytkownika
  - Listy
  - Tabele
  - Przyciski
  - Pola formularzy
  - Przeglądarka internetowa
  - itp.

Framework aplikacji: Content Providers

- Pozwalają wielu aplikacjom na korzystanie z tych samych danych, np. listy kontaktów.
- Nie są potrzebne, gdy do informacji ma mieć dostęp tylko jedna aplikacja – wówczas wystarczy utworzyć bazę danych i używać jej bez pośrednika.
- Są podklasami abstrakcyjnej klasy android.content.ContentProvider.
- Udostępniają funkcje query(...), insert(...), update(...), delete(...), getType(...).

Framework aplikacji: Resource Manager

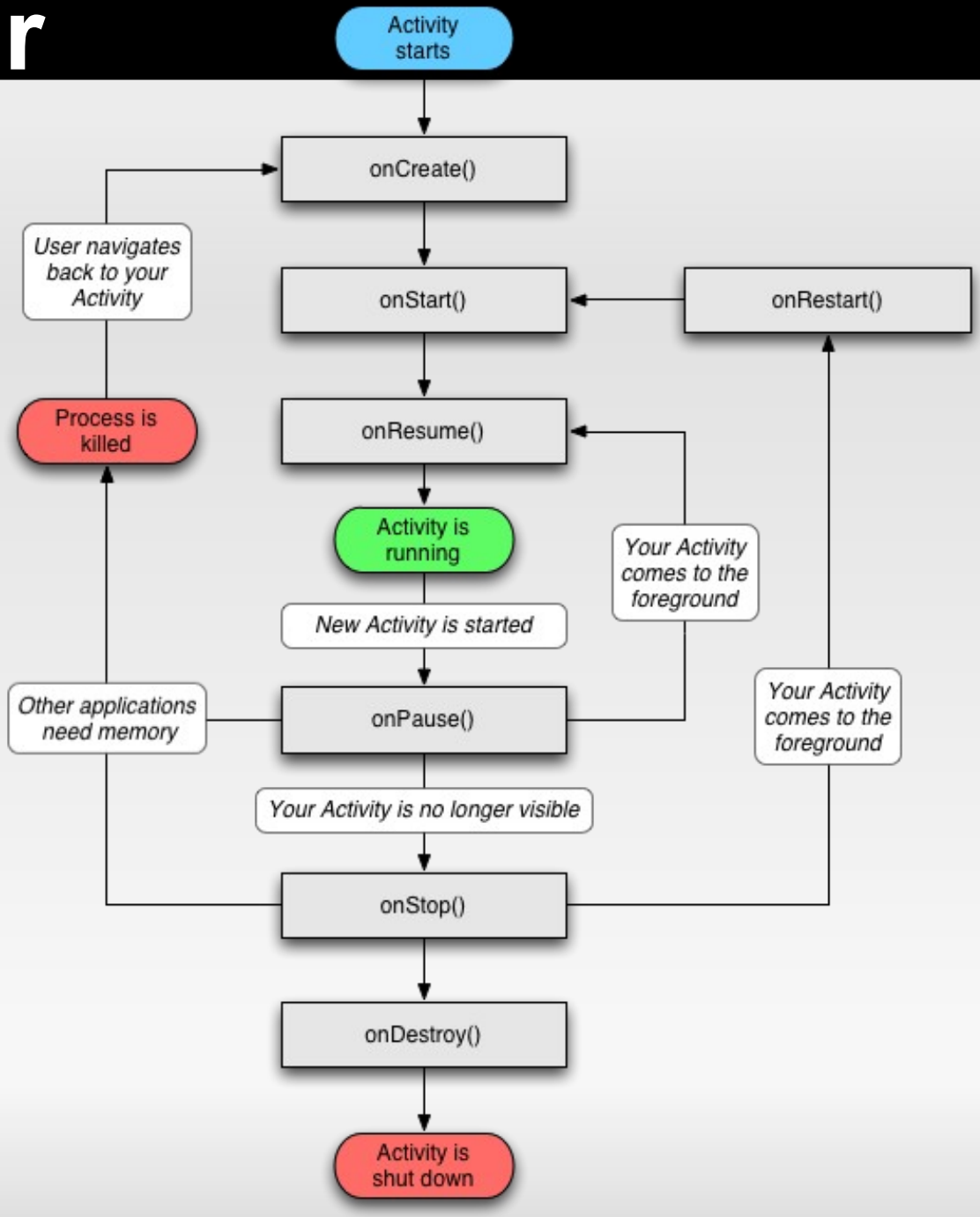
- Zarządza zewnętrznymi plikami, które nie są kodem, ale są potrzebne podczas kompilacji aplikacji.
  - Obrazy
  - Layouty (zapis wyglądu fragmentu okna w XMLu)
  - Stringi (np. dla aplikacji wielojęzycznych potrzebny jest komplet komunikatów dla każdego języka)
  - Kolory
  - itp.

Framework aplikacji: Notification Manager

- Służy do powiadamiania użytkownika o zdarzeniach, które zaszły „w tle”.
  - Wyświetlanie ikony w pasku stanu
  - Zapalanie diodek LED
  - Włączanie wibracji
  - Migotanie podświetleniem ekranu
  - Sygnały dźwiękowe

Framework aplikacji: Activity Manager

- Rozpoczyna, wstrzymuje, wznawia wykonanie procesów.
- Monitoruje ich aktywność (np. czy są widoczne).
- Może killować procesy w sytuacji braku pamięci operacyjnej.



Dalvik – maszyna wirtualna

- Przeznaczony dla urządzeń o ograniczonej pamięci i mocy obliczeniowej.
- Umożliwia uruchamianie programów napisanych w Javie i skonwertowanych do formatu **.dex** (Dalvik Executable).
  - Mniejszy rozmiar pliku niż **.jar**.
- Maksymalnie odchudzony, by zajmował jak najmniej pamięci RAM.
- Używa własnego bajtkodu.

Dalvik – maszyna wirtualna

- Brak możliwości kompilacji w locie (just-in-time compilation).
  - Kompilacja w locie – kompilacja bajtkodu do kodu maszynowego bezpośrednio przed wykonaniem danego fragmentu programu.
  - Technikę tę wykorzystuje współcześnie większość maszyn wirtualnych Javy.
- Stworzony z myślą o równoległej pracy wielu maszyn wirtualnych bazującej na wsparciu ze strony systemu operacyjnego.

Bezpieczeństwo

- Podpisywanie pakietów .apk.
- Mechanizm zezwoleń (*permissions*) przyznawanych każdemu pakietowi.
- Wykorzystanie user ID nadawanego pakietom do izolacji aplikacji.
- Mechanizm przekazywania zezwoleń na dostęp do konkretnego URI (konkretnego fragmentu danych) między aplikacjami.

Podpisywanie pakietów

- Każdy pakiet .apk jest podpisany certyfikatem, do którego klucz prywatny posiada twórca pakietu.
- Podpis pozwala na rozpoznanie, czy dwa pakiety pochodzą z tego samego źródła.
- Tylko pakiety z identycznymi podpisami mogą być traktowane jako jedna aplikacja i współdzielić np. zezwolenia czy ID użytkownika.

Zezwolenia

- Domyślnie aplikacji nie wolno wykonać żadnej operacji, która mogłaby mieć wpływ na inną aplikację, system operacyjny lub użytkownika.
  - W tym np.: odczyt prywatnych danych, dostęp do sieci, modyfikacja plików innych aplikacji.
- Aby uzyskać dane zezwolenie, należy je zadeklarować w pliku AndroidManifest.xml, który jest częścią pakietu .apk.
- Można również utworzyć własne zezwolenie.

Zezwolenia

- Zezwolenia są przyznawane pakietowi statycznie w momencie instalacji.
  - Automatycznie – analizując zezwolenia innych pakietów.
  - Ręcznie – przez pytanie do użytkownika.
  - Z raz przyznanego zezwolenia aplikacja może korzystać kiedy zechce bez pytania użytkownika.
  - Jeżeli aplikacja nie otrzymała zezwolenia w czasie instalacji, to próba skorzystania z niego zawsze zakończy się niepowodzeniem, o którym użytkownik nie jest powiadamiany.

ID użytkownika

- W momencie instalacji każdy pakiet .apk otrzymuje unikalny numer user ID.
- Jądro Linuksa izoluje je od siebie.
- Kod z dwóch pakietów o różnym user ID nie może być wykonywany w jednym procesie.
- W momencie instalacji pakiety z tym samym podpisem mogą prosić o przydzielenie wspólnego user ID. Są wtedy traktowane jako jedna aplikacja i mają wspólną listę zezwoleń.

Dostęp do plików

- Plik dziedziczy user ID swojego właściciela po aplikacji, która go utworzyła.
- Domyślnie pliki mogą być odczytywane i zapisywane jedynie przez tę aplikację.
- Aby plik mógł być współdzielony z innymi aplikacjami, trzeba w momencie tworzenia ustawić odpowiednie flagi:
  - MODE_WORLD_READABLE
  - MODE_WORLD_WRITEABLE

Zezwolenia URI

- *Uniform Resource Identifier* – standard internetowy umożliwiający łatwą identyfikację zasobów w sieci.
- Jest wykorzystywany w Androidzie do wysyłania zapytań do Content Providerów.
  - content://contacts/people/
    – lista wszystkich zapisanych kontaktów
  - content://contacts/people/23
    – rekord z danymi osoby o id równym 23

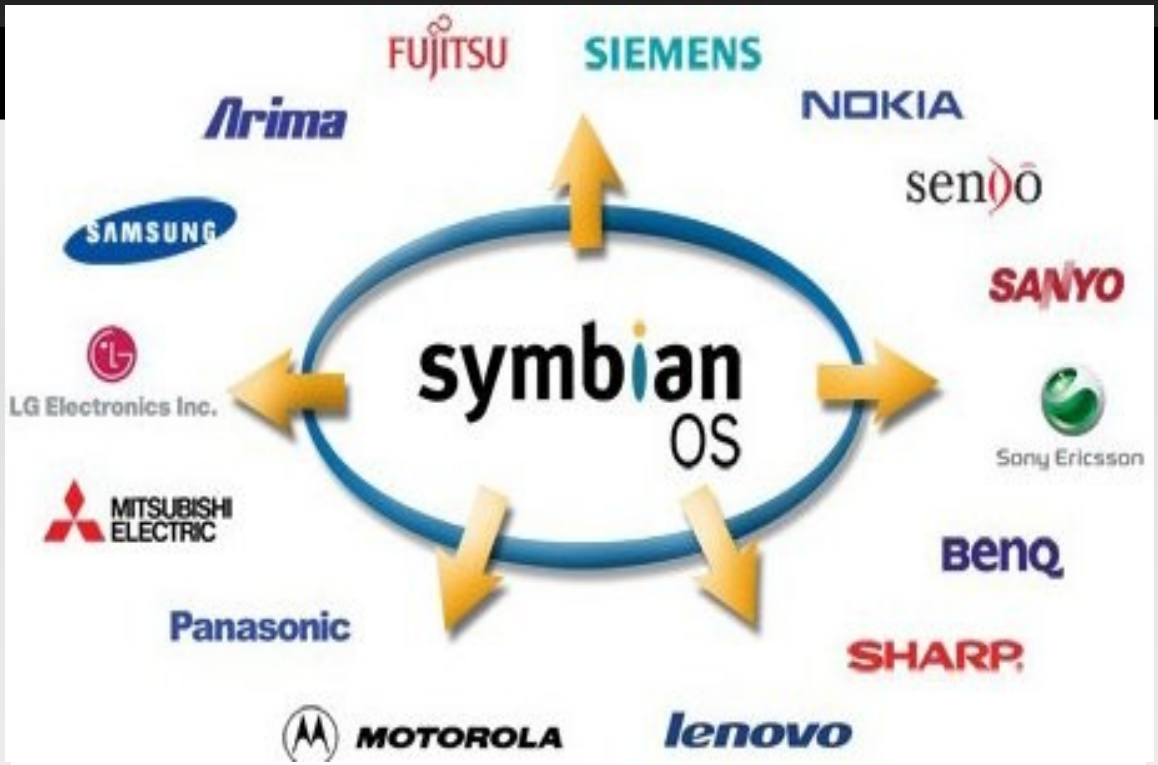
Zezwolenia URI - przykład

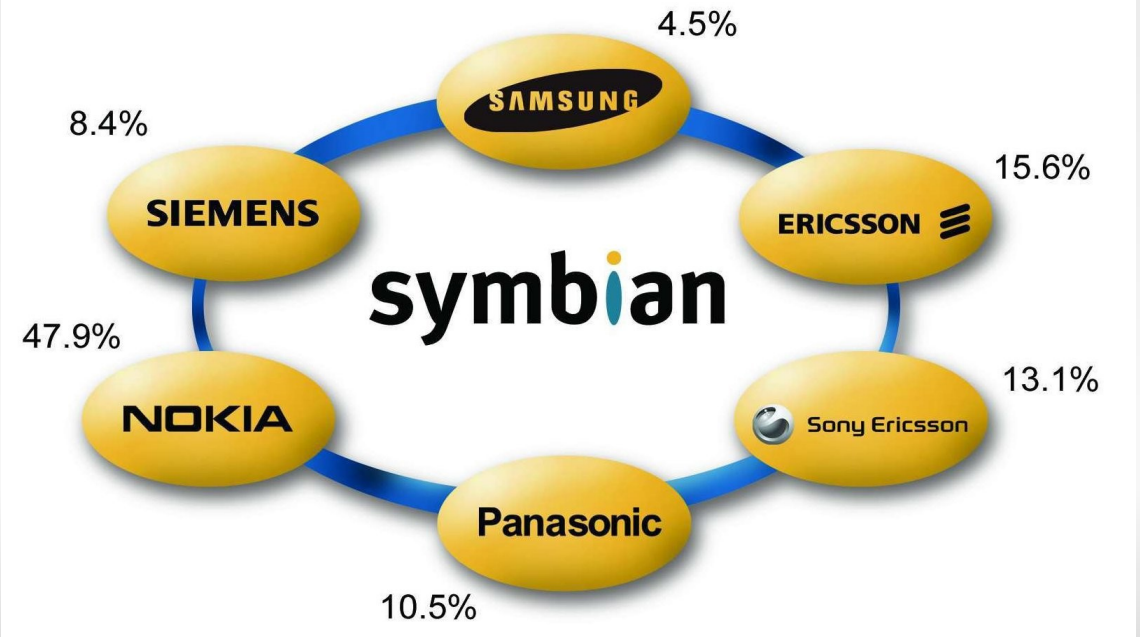
- Dostęp do poczty e-mail wymaga zezwolenia.
- Takie zezwolenie posiada program pocztowy.
- Przypuśćmy, że chcemy otworzyć zdjęcie z załącznika do e-maila.
- Przeglądarka obrazków nie ma dostępu do załącznika, bo nie ma zezwolenia na dostęp do poczty, a załącznik jest w poczcie.
- Zatem program pocztowy przekazuje przeglądarce obrazków zezwolenie na dostęp do konkretnego zasobu – zdjęcia w załączniku.


symbian
OS

Symbian

- Konsorcjum Symbian Ltd. powstało 24.06.98
- W jego skład weszły między innymi Nokia, Samsung, Motorola, Siemens i Sony Ericsson.
- Dziesięć lat później, Nokia przejęła kontrolę nad Symbian Ltd.





Symbian

- Symbian OS został stworzony w oparciu o system EPOC firmy Psion PLC, który powstał pod koniec lat 80'.
- Symbian działa wyłącznie na procesorach ARM.

  (98% wszystkich telefonów komórkowych – dane z 2007).
- Symbian posiada 59% rynku smartfonów (dane z połowy 2008 roku). To ponad czterokrotnie więcej niż największy konkurent.

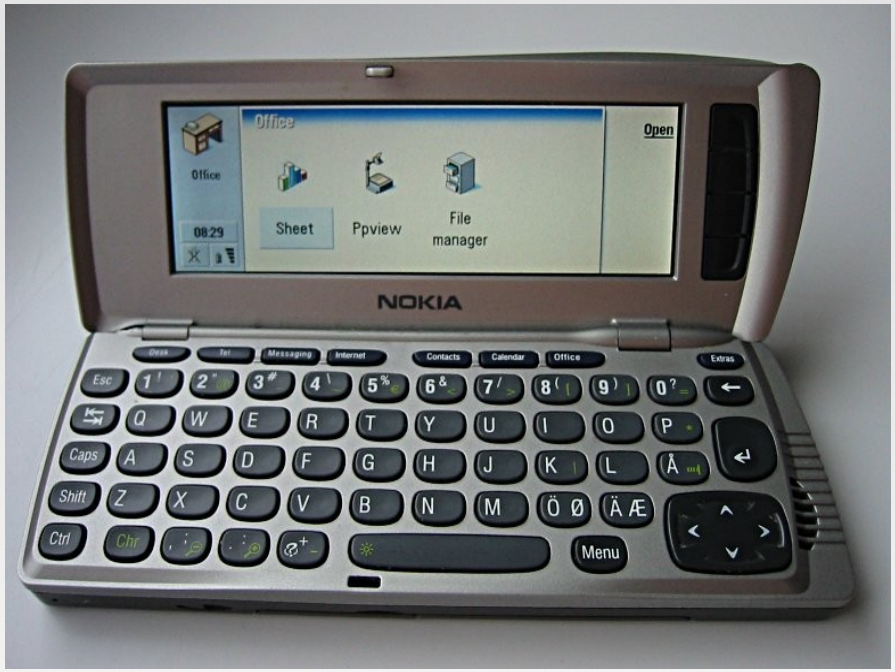
Symbian na przykładach

Nokia 9210 Communicator

Pierwszy telefon z Symbianem

Symbian OS v6.0

Rok produkcji: 2000



Symbian na przykładach

Sony Ericsson P800

Platforma UIQ na Symbian OS v7.0

Rok produkcji: 2003

Symbian na przykładach

Nokia 7610

Platforma S60 na Symbian OS v7.0

Rok produkcji: 2003

Symbian na przykładach

Nokia N90

Platforma S60 2nd Edition

na Symbian OS v8.1

Rok produkcji: 2005

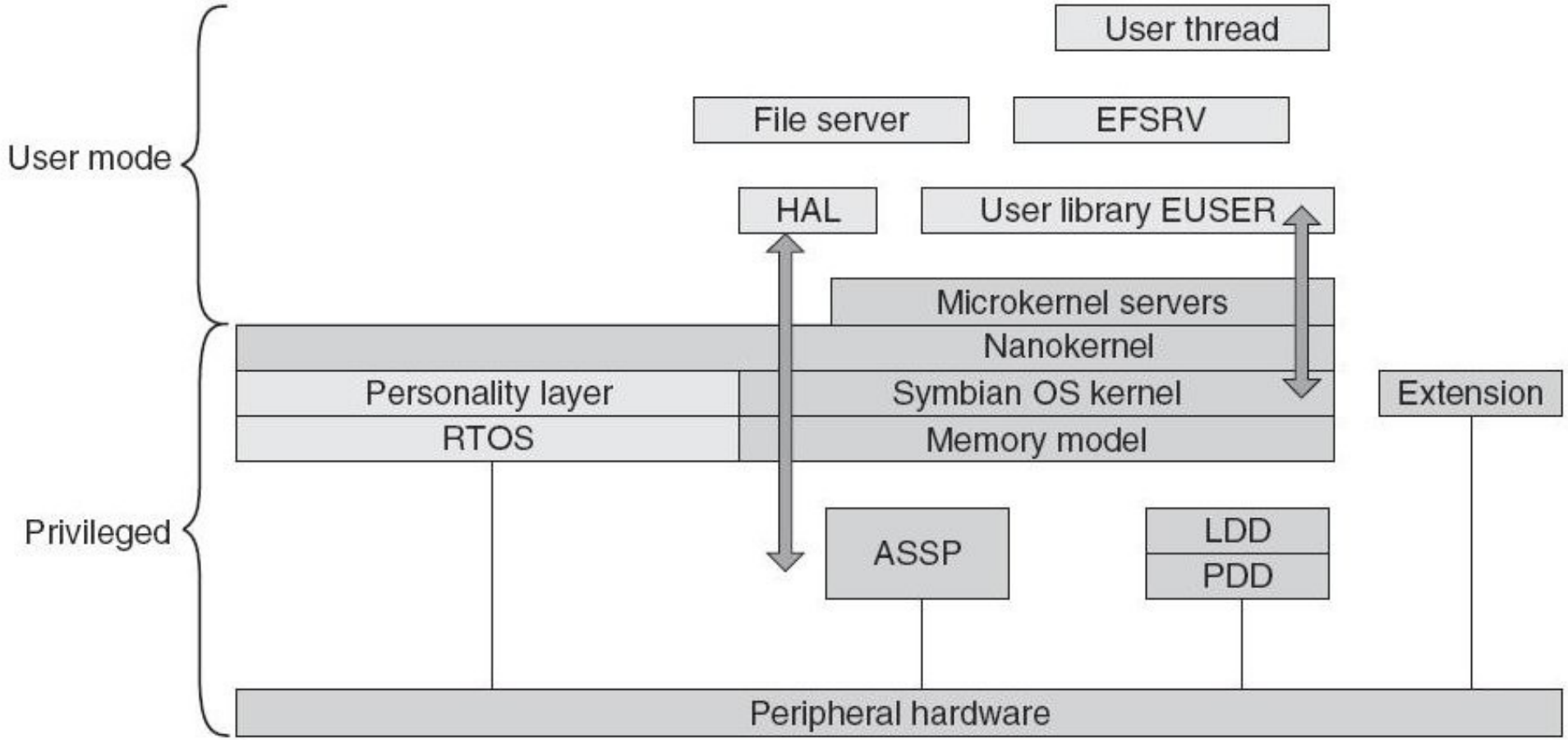
Symbian na przykładach

Nokia E66

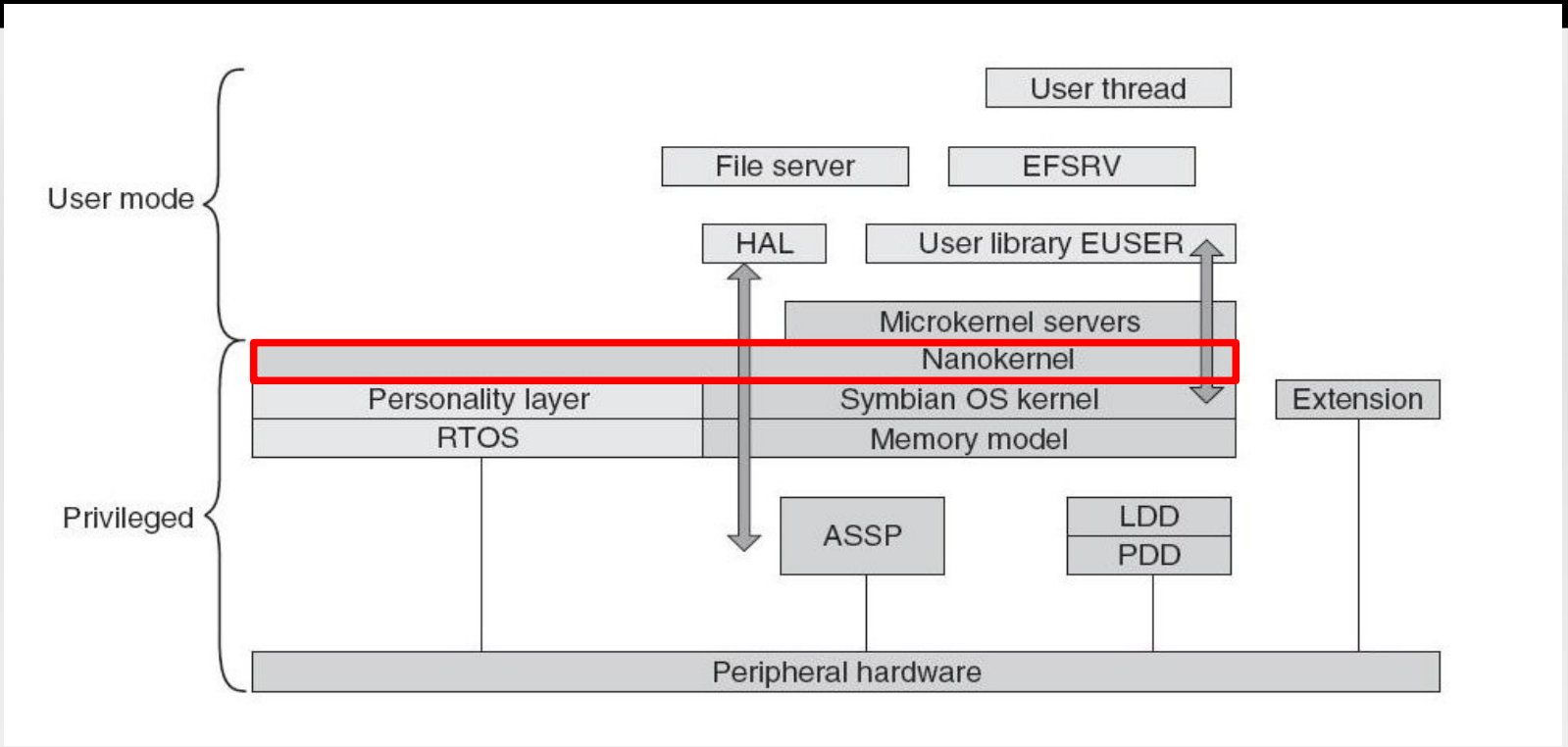
Platforma S60 3rd Edition na Symbian OS v9.2

Rok produkcji: 2008

Struktura jądra

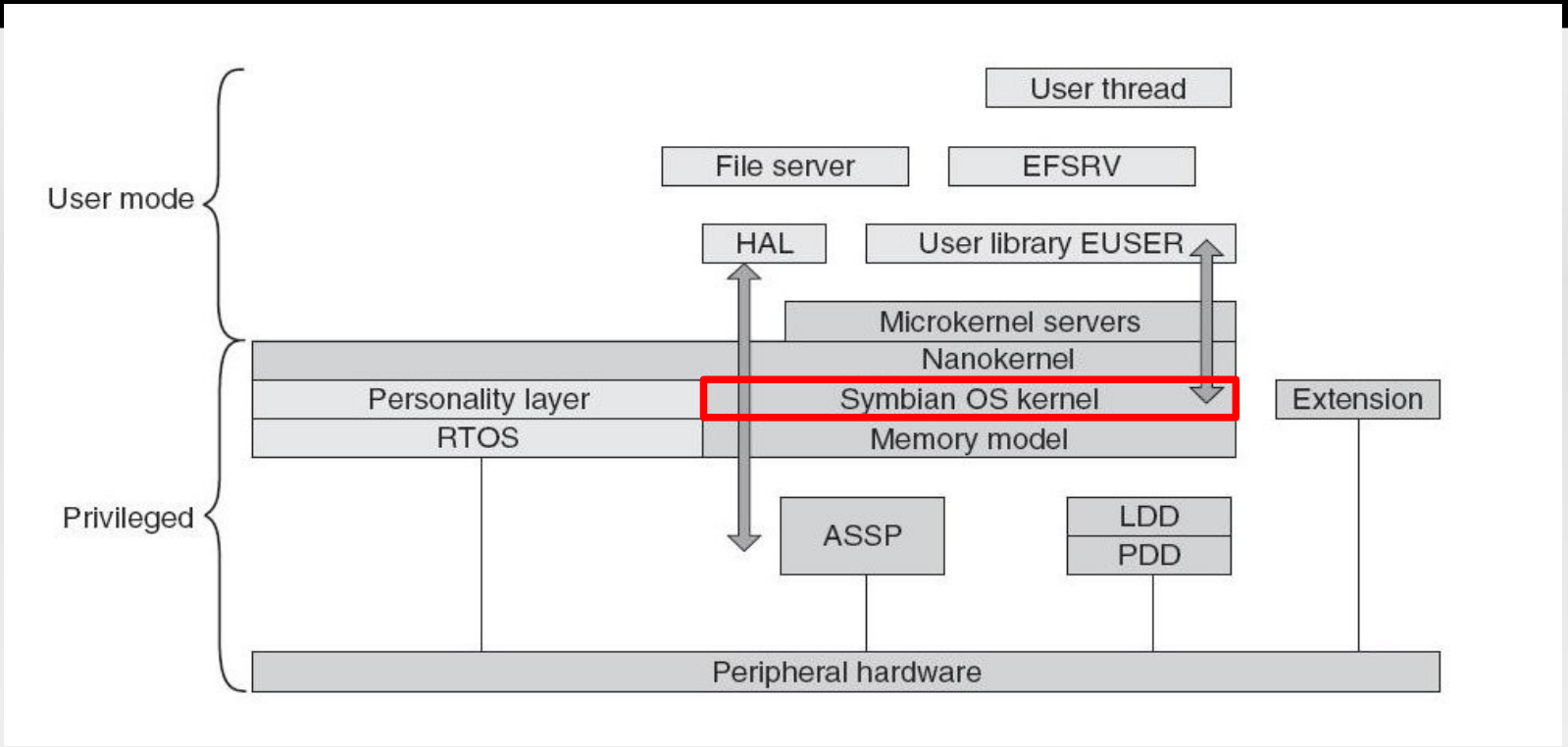


Jądro - Nanokernel



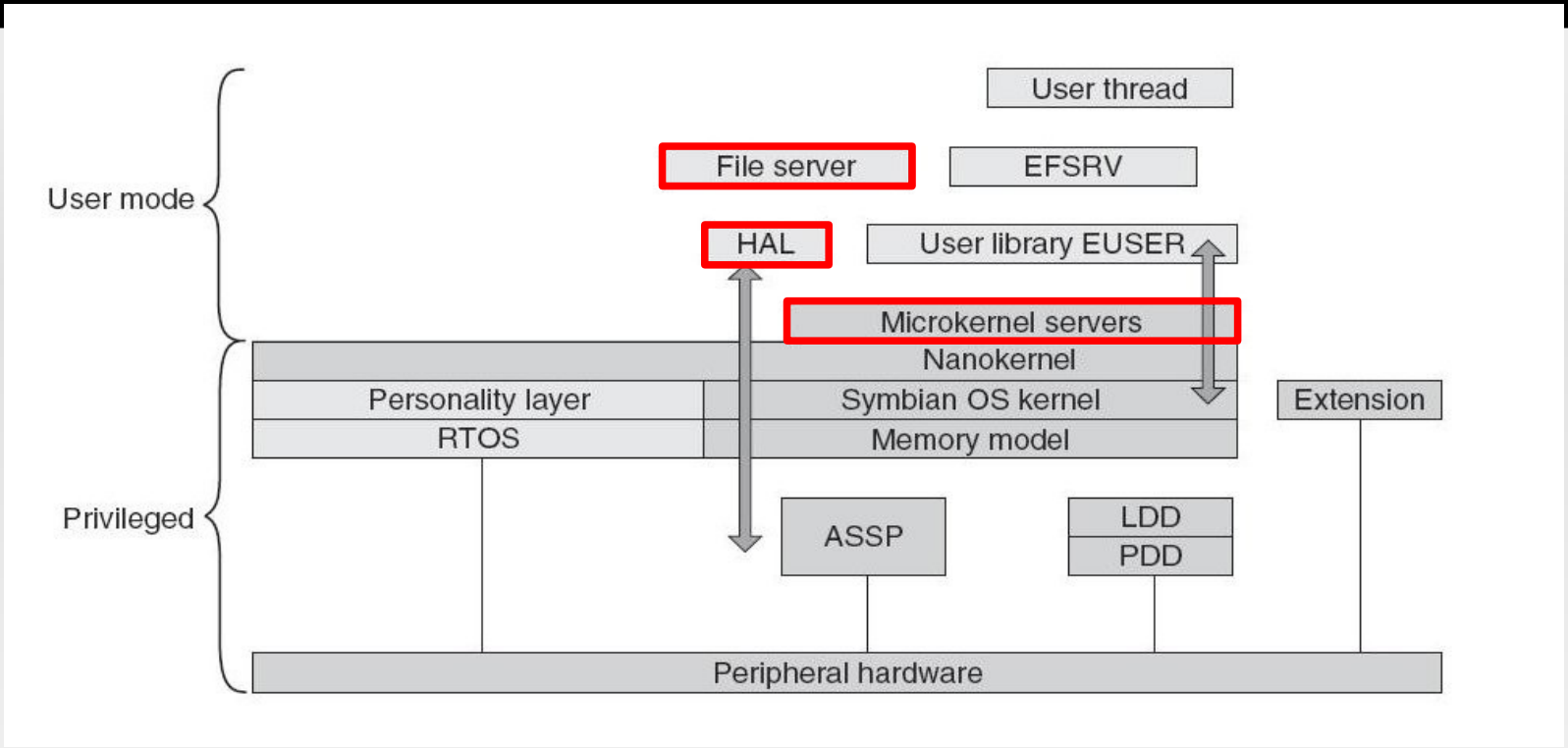
- Obsługuje najprostsze i najbardziej niskopoziomowe operacje o niewielkim stopniu skomplikowania.
- Jest wykorzystywany przez Symbian OS kernel oraz RTOS.

Jądro – Symbian OS kernel



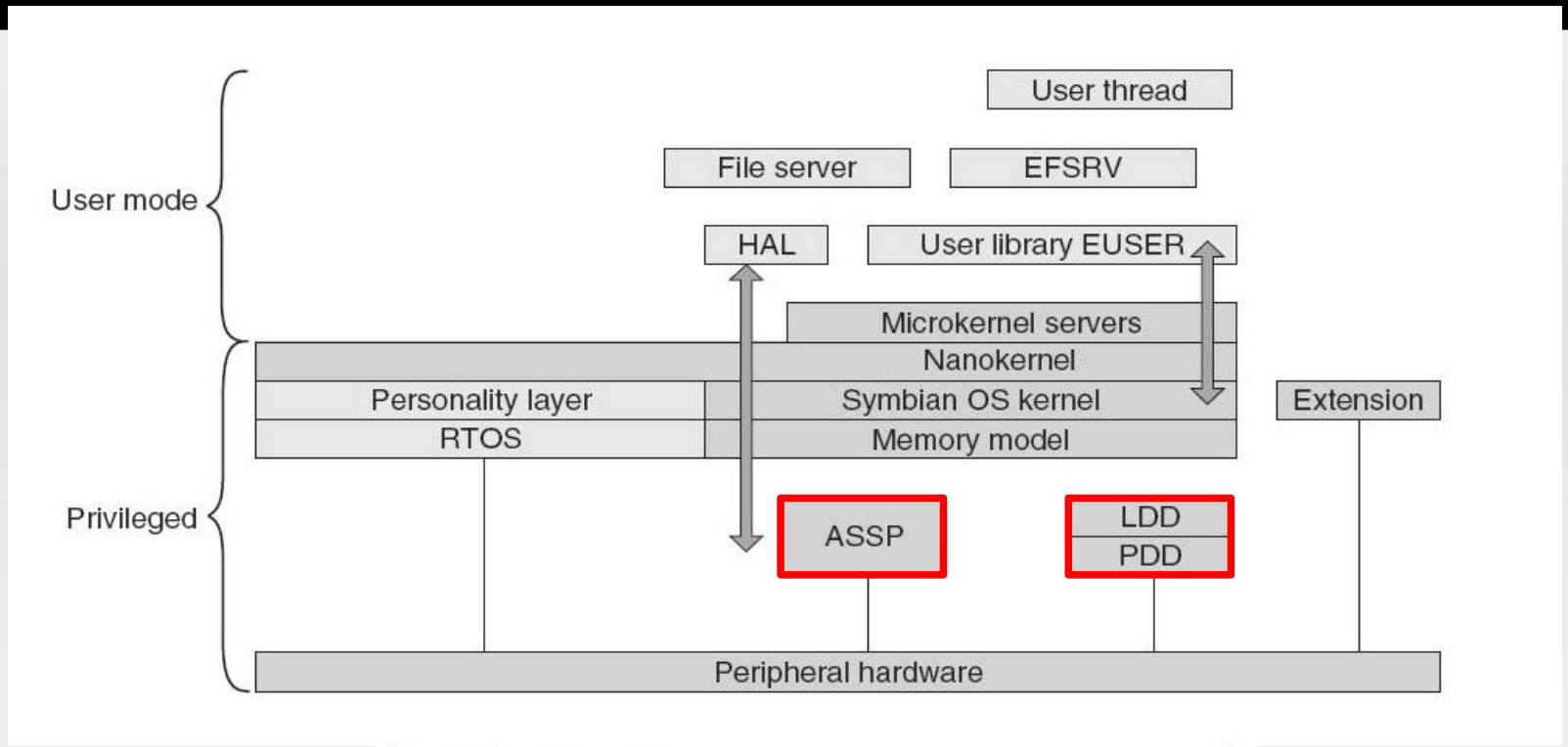
- Najniższa warstwa dostępna dla aplikacji z User mode.
- Udostępniane funkcje stanowią w dużym uproszczeniu kombinacje prostszych funkcji nanojądra.
- Zajmuje się takimi kwestiami jak: dynamiczne przydzielanie pamięci, ładowanie dynamicznych bibliotek, zarządzanie procesami, przełączanie kontekstów, komunikacja międzyprocesowa itp.
- Wszystkie zadania realizowane przez Symbian OS kernel (w tym przełączanie kontekstów!) są w pełni wywłaszczalne.

Jądro – Microkernel servers



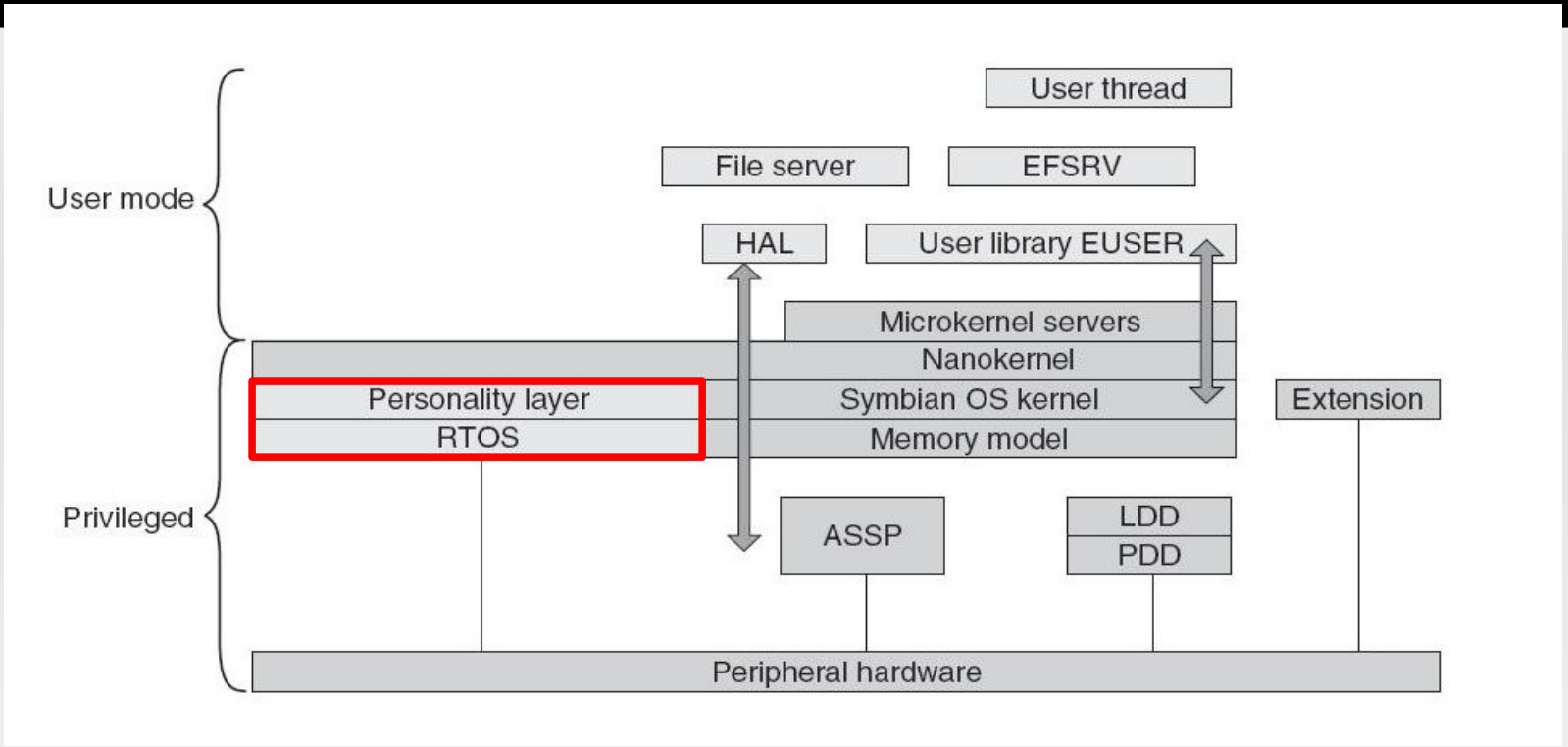
- Typowe dla architektury mikrojądra wyniesienie wielu usług poza przestrzeń jądra.

Jądro - Sterowniki



- LDD – Logical Device Drivers są pluginami do Symbian OS kernel.
- PDD – Physical Device Drivers są pluginami do LDD.
- ASSP – Application-specific standard product – zestaw sterowników dla standardowych urządzeń.
- Wszystkie sterowniki są ładowane dynamicznie ze względu na oszczędność zasobów systemu.

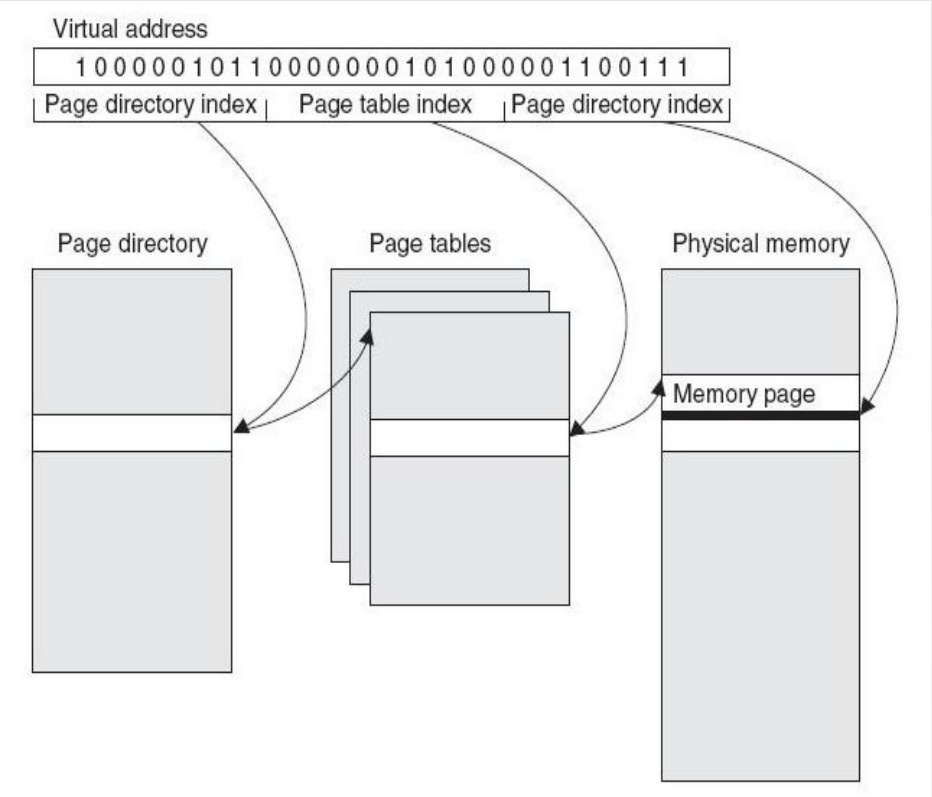
Jądro – Real-time OS i Personality Layer



- RTOS obsługuje protokoły telefoniczne np. GSM, 3G jak również Bluetooth, IrDA.
- Personality layer stanowi „tryb” real-time dla aplikacji z user space, jak i warstwę tłumaczącą protokół używany przez aplikację na protokół hardware'u telefonu.

Zarządzanie pamięcią

- Strony wielkości 4KB (zazwyczaj).
- Dwupoziomowa tablica stron.
- Procesory ARM posiadają zaawansowane MMU z TLB, które zwiększa szybkość translacji adresów.



Procesy - Scheduling

- Procesy real-time są wykonywane przed innymi procesami.
- Procesy real-time mające swoje deadline'y są wykonywane w kolejności deadline'ów.
- Procesy real-time nie posiadające deadline'ów dostają pewien ułamek czasu procesora.
- Strategia nie optymalna, ale dość dobra.
- Istnieją bardziej złożone strategie dające lepsze wyniki, ale są znacznie bardziej skomplikowane i nie do końca potrzebne.
- 64 priorytety (zależne od bliskości deadline'u) na 64 kolejkach.

Systemy plików

- Ze względu na prostotę i potrzebę wymiany danych z innymi systemami, Symbian używa wewnętrznie FAT16.
- FileServer stanowi warstwę abstrakcji podobną do Unix'owego VFS i może być używany z większością systemów plików.
- Istnieją również implementacje zdalnych systemów plików (NFS i SMB) dla Symbian OS.

Bezpieczeństwo

- Pierwszy wirus na Symbian OS pojawił się w 2004 roku. Caribe poza rozsyłaniem się przy użyciu protokołu Bluetooth był nieszkodliwy. Zarysował jednak problem bezpieczeństwa urządzeń typu PDA i smartfonów.
- W Symbian OS v8.x i wcześniejszych został zaimplementowany mechanizm dający użytkownikowi wybór rodzaju uprawnień udzielanych poszczególnym zainstalowanym aplikacjom.

Symbian Signed

- W momencie pojawienia się na rynku Symbian OS v9.0 konsorcjum Symbian Ltd. wprowadziło na rynek kontrowersyjny program Symbian Signed, dzięki któremu użytkownik został zwolniony z obowiązku decydowania o przywilejach instalowanych aplikacji.
- Od tej pory każda aplikacja chcąca skorzystać z usług jądra w sposób który umożliwia zaburzenie jego pracy musi zostać uprzednio podpisana.
- Aby dostawca oprogramowania mógł zaoferować podpisane oprogramowanie, musi zarejestrować się w programie Symbian Signed oraz wysłać stworzony software do niezależnej instytucji testowej. Instytucja testowa sprawdza czy opis funkcjonalności oprogramowania zgadza się z jej kodem oraz czy wszystkie dostępy do usług jądra zostały udokumentowane.
- Można znaleźć w internecie strony, które umożliwiają podpisanie aplikacji z dowolnymi wybranymi uprawnieniami, jak i programy umożliwiające podpisanie aplikacji pod konkretny telefon (IMEI).

Symbian Signed

Zalety:

- Symbian jest bardzo bezpiecznym systemem. Jedynie sprawdzone aplikacje od wykwalifikowanych dostawców mogą uzyskać dostęp do zaawansowanych funkcji systemu.

Wady:

- Utrudnione tworzenie aplikacji OpenSource i przez małe firmy.
- Brak możliwości stworzenia dla swojego telefonu aplikacji o pełnych uprawnieniach.

Pisanie aplikacji dla Symbian OS

- Głównym językiem jest C++ odbiegający nieznacznie od standardu, choć można pisać również w takich językach jak: OPL, Python, Visual Basic, Simkin, Perl, Java ME, PersonalJava.
- Niestety pisanie aplikacji dla Symbian OS jest bardzo trudne – wymaga używania specjalnych technik stworzonych pod hardware z lat '90, z których płyną wątpliwe korzyści, a które zmuszają programistę do skupiania się na niskopoziomowych aspektach zamiast na logice aplikacji.

Pisanie aplikacji dla Symbian OS

- Istnieją SDK zawierające potrzebne biblioteki, dokumentację oraz kompilatory dla wielu platform bazujących na Symbianie.
- SDK zawiera również emulator Symbian OS umożliwiający testowanie stworzonej aplikacji w środowisku zbliżonym do smartfona z użyciem debuggera.

Pisanie aplikacji dla Symbian OS

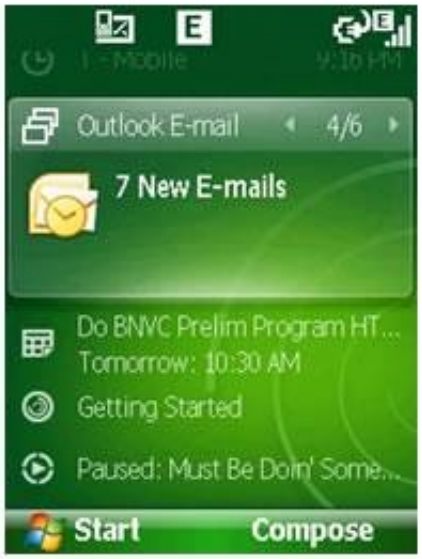
- Aplikacje pod Symbian OS najczęściej są tworzone przy użyciu specjalnie do tego przeznaczonych IDE. Aktualnie sugerowane przez producenta jest Carbide.c++ autorstwa firmy Nokia zbudowanego na bazie Eclipse. Posiada cztery wersje różniące się możliwościami. Prostsze są darmowe, natomiast bardziej rozbudowane są płatne.
- Istnieje plugin dla Microsoft Visual Studio 2003 i 2005 o nazwie carbide.vs, umożliwiający pracę nad aplikacjami dla Symbian OS pod tymi środowiskami.

Pisanie aplikacji dla Symbian OS

- Gotowe aplikacje pakowane są w pliki SIS, które można w dowolny sposób umieścić w telefonie (połączenie z PC, Bluetooth, karty pamięci itd.).
- Należy pamiętać, że od wersji Symbian OS 9.x, aby aplikacja posiadała istotne prawa w systemie, musi być podpisana w programie Symbian Signed.
- Inną przeszkodą jest konieczność utrzymywania różnych wersji aplikacji pod różne platformy bazujące na Symbian OS (S60, UIQ, MOAP).

Windows Mobile

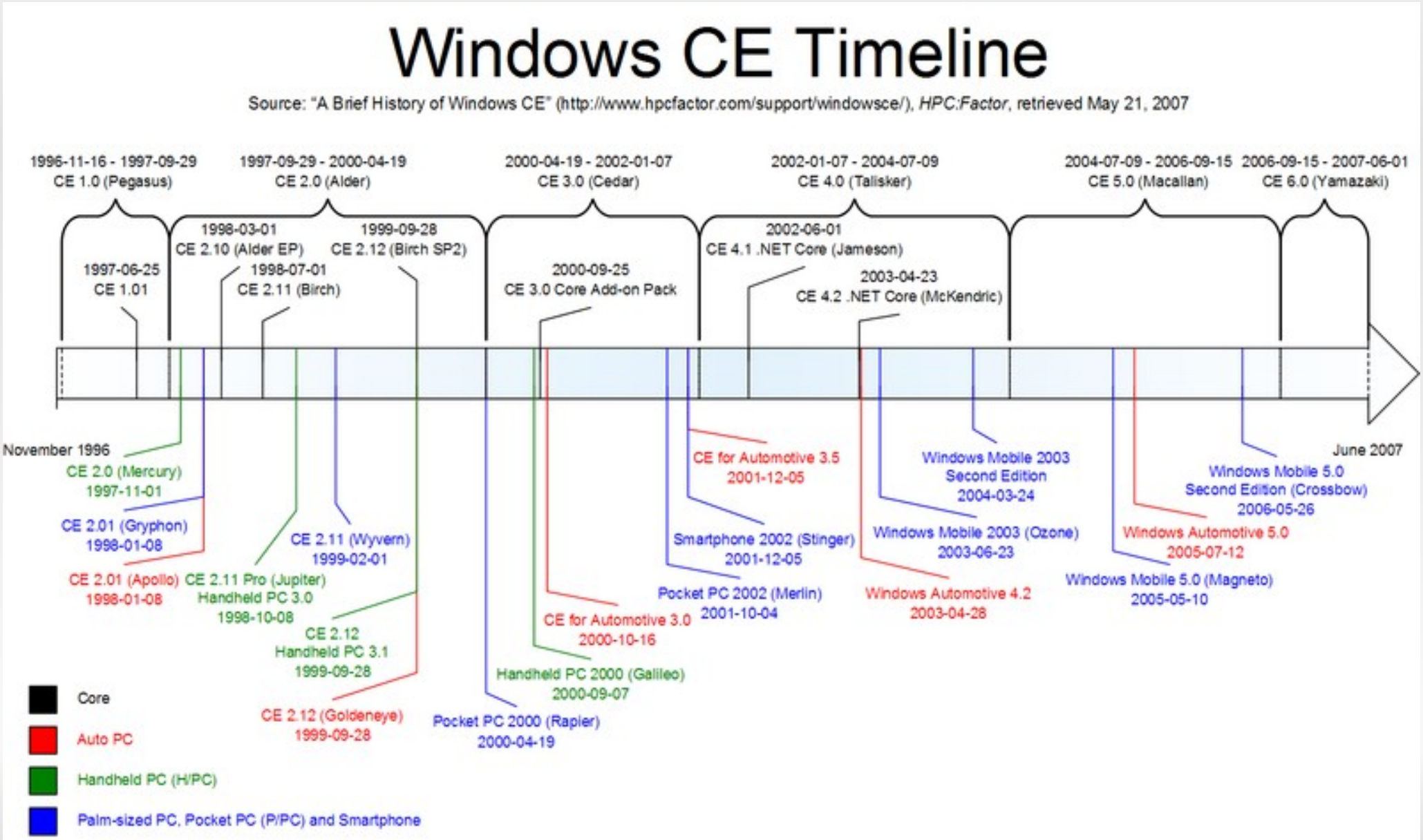
Windows Mobile



Windows Mobile – historia

- Wywodzący się z projektów *WinPad* i *Pulsar*, rozwijanych w latach 1992-1994, zaniechanych z powodu braku odpowiedniego sprzętu.
- Na ich podstawie w 1996 roku powstał pierwszy Windows CE – *Pegasus*.

Windows Mobile – historia



Windows Mobile – sprzęt

- Microsoft definiował konkretne wymagania dla urządzeń.
  - Palm-size PC – procesory SH3 i MIPS, najstarsza klasa urządzeń pod Windows CE 2.
  - Handheld PC – rozdzielczość ekranu nie mniejsza niż 480x240, klawiatura, slot kart CF, PCIMCIA, IRDA, łączność przez port szeregowy lub USB.
  - Pocket PC – ekran dotykowy, dżojstik kierunkowy, procesor ARM, Intel XScale, MIPS albo SH3 (później ujednolicone do ARM).

Windows Mobile – sprzęt

- Smartphone – telefony z trochę niższej półki niż Pocket PC. Brak ekranu dotykowego, niższa rozdzielczość ekranu.
- Portable Media Center – odtwarzacze muzyki, licencjonowanie zarzucone na rzecz microsoftowego projektu Zune.
- Auto PC – obsługa GPS, interfejs USB do odtwarzania muzyki, bluetooth. Procesor ARM co najmniej 300 MHz.

Windows Mobile – pisanie aplikacji

- Natywny kod pisany w C++.
- Kod w .NET Compact Framework, C# lub Visual Basic .NET – specjalna wersja frameworku .NET dla aplikacji mobilnych.
- Aplikacje webowe po stronie serwera (ASP .NET Mobile Controls i Ajax), odpalane w przeglądarce internetowej.

Windows Mobile – pisanie aplikacji

- Microsoft dostarcza Software Development Kits do Visual Studio.
- Dla wersji wcześniejszych niż Windows Mobile 2003 używano eMbedded Visual Tools.
- Na SDK składają się:
  - Dokumentacja API i przykłady
  - Biblioteki
  - Obrazy urządzeń na emulator

Windows Mobile – podsumowanie

- Plusy:
  - Łatwy w obsłudze – przyzwyczajenie większości użytkowników do Windowsa.
  - Możliwa synchronizacja z innymi produktami Microsoftu.
  - Duża baza programów.
- Minusy:
  - Duże różnice pomiędzy różnymi urządzeniami, słaba przenośność programów.
  - Duży i ciężki system.

iPhone OS


AT&T
00:00
Text
Thursday
27
Calendar
Photos
Camera
YouTube
Stocks
280
Maps
23°
Weather
Clock
Calculator
Notes
Settings
iTunes
App Store
Phone
Mail
Safari
iPod

iPhone OS

- System operacyjny na telefony Apple – iPhone i iPhone Touch.
- Tak jak Mac OS X bazuje na systemie Darwin.
  - Zgodny z POSIXem.
  - OpenSource rozwijany przez Apple.
  - Jądro hybrydowe.

iPhone OS

- Cztery warstwy systemu:
  - Core OS Layer – TCP/IP, sockety, zarządzanie energią, system plików.
  - Core Services Layer – wątki, baza SQLite.
  - Media Layer – audio i video, formaty obrazów.
  - Cocoa Touch Layer – interfejs użytkownika, obsługa kamerki.

iPhone OS – aplikacje

- Autoryzowane aplikacje do ściągnięcia przez *Apple App Store.*
- Aplikacje webowe dostępne przez przeglądarkę *Safari.*
- Brak wsparcia dla *Javy* oraz *Flasha.*

iPhone OS – aplikacje

- SDK dla iPhone jest darmowe.
- Ale aby opublikować aplikację trzeba zarejestrować się w iPhone Developer Program – co kosztuje $99-299.
- Dystrybucja przez App Store – samemu można ustalić cenę, z której otrzyma się 70%.
- Dostępne są sposoby hakowania iPhone'a, aby zainstalować na nim aplikacje spoza App Store.

iPhone OS – podsumowanie

- Plusy:
  - Apple'owy design.
- Minusy:
  - Restrykcyjny system dystrybucji aplikacji.
  - Brakuje podstawowych funkcjonalności – np. Javy, Flasha.

BlackBerry

BlackBerry

BlackBerry

- System operacyjny na urządzenia BlackBerry firmy Research in Motion.
- Popularny w firmach, ponieważ umożliwia automatyczną synchronizację z firmowym serwerem BlackBerry Enterprise Server poprzez sieci komórkowe.
- Komunikacja z serwerem BES jest szyfrowana za pomocą Triple DES / AES.
- Providerzy zazwyczaj oferują zryczałtowaną stawkę za nielimitowaną komunikację pomiędzy serwerem a urządzeniem.

BlackBerry

- Samo urządzenie ma ograniczone możliwości, bazuje głównie na komunikacji z serwerem.
- Niewielki jest wachlarz dodatkowych aplikacji.
- BlackBerry przez większość użytkowników postrzegany jest jako „urządzenie do obsługi poczty e-mail”.

Palm OS

Palm OS


1:50 pm
All
Email
GoogleMaps
HotSync
IM
Memos
Messaging
On Demand
Phone
Pics&Videos
Prefs
pTunes
ScreenShot

Palm OS

- System operacyjny na maszyny Palm (aczkolwiek część z nich działa również na Windows Mobile).
- Obecnie, po przejęciu od Palm Inc. przez Access Co., nosi nazwę *Garnet OS.*

Palm OS

- Do wersji 5.0 na architekturę Motorola/FreeScale Dragonball, później na architekturę ARM.
- System jednozadaniowy.
- Rozdzielczości ekranu do 480x320.
- Rozpoznawanie pisma systemem *Graffiti 2*.

Palm OS – tworzenie aplikacji

- Istnieje ponad 50 000 zewnętrznych aplikacji stworzonych na Palm OSa.
- Możliwość programowania w C/C++ za pomocą komercyjnej platformy CodeWarrior Development Studio (już nie rozwijane) lub darmowego prc-tools (oparte na gcc).
- Można też pisać w C na samym Palmie, za pomocą kompilatora OnBoardC.

Palm OS – tworzenie aplikacji

- Można developować w innych językach i na innych platformach takich jak:
  - PocketC/PocketC Architect
  - CASL (Compact Application Solution Language)
  - AppForge Crossfire (VB, VB .NET, C#)
  - Handheld Basic, Pendragon Forms, Satellite Forms (VB podobne)
  - Pascal – PP Compiler (na samym Palmie), Pocket Studio (Delphi-podobna platforma pod Windowsa)
  - LispMe – implementacja Scheme

Koniec

Pytania?

Bibliografia

- http://code.google.com/intl/pl/android/
- http://www.android.com/
- http://www.dalvikvm.com/
- http://www.openhandsetalliance.com/
- http://blogs.zdnet.com/Burnette/?p=515
- Michael J. Jipping. *Smartphone Operating System Concepts with Symbian OS. A tutorial Guide.*

Bibliografia

- http://searchmobilecomputing.techtarget.com/ge
- http://msdn.microsoft.com/pl-pl/windowsmobile/b
- http://msdn.microsoft.com/pl-pl/windowsmobile/b
- http://www.hpcfactor.com/support/windowsce/
- http://msdn.microsoft.com/pl-pl/windowsmobile/b
- http://cdecas.free.fr/computers/pocket/simon.php
- Wikipedia