VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

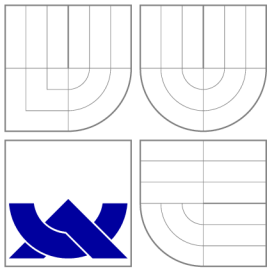
DEMONSTRAČNÍ APLIKACE SYMBIAN S60

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

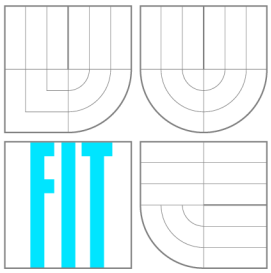
AUTOR PRÁCE
AUTHOR

MAREK HLOBIL

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

DEMONSTRAČNÍ APLIKACE SYMBIAN S60

DEMO APPLICATION FOR SYMBIAN S60

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — MAREK HLOBIL
AUTHOR

VEDOUCÍ PRÁCE — Ing. PETR CHMELAŘ
SUPERVISOR

BRNO 2009

Abstrakt

Tato bakalářská práce se zabývá operačním systémem Symbian S60 pro mobilní telefony. Je zde popsána struktura operačního systému Symbian a rozebrány jsou jeho stěžejní části. Hlavním cílem práce je demonstrace možností tohoto operačního systému na příkladě. Pro tento účel byla vytvořena aplikace v programovacím jazyce Python, která filtruje zprávy SMS na základě uživatelem definovaných pravidel.

Abstract

The bachelor's thesis concerns with Symbian S60 operating system for mobile phones. The structure and properties of Symbian OS is described in the thesis. The aim of the thesis is to demonstrate capabilities of Symbian S60 by design of a demonstration application. For this purpose, an incoming SMS filter was implemented in Python programming language.

Klíčová slova

Symbian S60, EPOC, operační systém, Python, MIDP, CLDC, pys60, SMS, filter

Keywords

Symbian S60, EPOC, operating system, Python, MIDP, CLDC, pys60, SMS, filter

Citace

Marek Hlobil: Demonstrační aplikace Symbian S60, bakalářská práce, Brno, FIT VUT v Brně, 2009

Demonstrační aplikace Symbian S60

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Petra Chmelaře.

.....
Marek Hlobil
20. května 2009

Poděkování

Děkuji panu Ing. Petru Chmelařovi, za jeho odbornou pomoc, rady a cenné připomínky, které mi pomohly při vypracování mé bakalářské práce.

Obsah

1 Úvod **2**

2 Symbian S60 **3**
2.1 Úvod . . . 3
2.2 Historie . . . 3
2.3 Struktura operačního systému . . . 4
2.3.1 Kernel – EKA2 . . . 4
2.3.2 Nanokernel . . . 5
2.3.3 Synchronizační prostředky . . . 6
2.3.4 Správa paměti, paměťový model . . . 6
2.3.5 File server . . . 9
2.3.6 Loader . . . 11
2.3.7 Window server . . . 12

3 Demonstrační aplikace **13**
3.1 Zadání . . . 13
3.2 Vývojové prostředky . . . 13
3.2.1 Open C a C++ . . . 14
3.2.2 Java . . . 14
3.2.3 Python . . . 15
3.2.4 Flash Lite . . . 16
3.2.5 Qt . . . 16
3.3 Návrh řešení . . . 17
3.4 Implementace . . . 18
3.4.1 Třída `SMSfilter` . . . 18
3.4.2 Třída `FileSel` . . . 21
3.4.3 Struktura konfiguračního souboru . . . 22
3.5 Používání programu, omezení . . . 22
3.6 Možná rozšíření . . . 23

4 Závěr **27**

A Obsah přiloženého CD **30**

Kapitola 1

Úvod

Symbian S60 je operační systém, jehož problematikou se ve své bakalářské práci zabývám. Téma mé práce zní „Demonstrační aplikace Symbian S60“ a zvolil jsem si jej proto, že i já jsem uživatelem tohoto operačního systému. Při jeho každodenním používání jsem si uvědomil, že postrádám některé funkce, které nejsou zajištěny prostřednictvím běžně dostupných programů. Proto jsem se rozhodl seznámit se blíže s prostředky operačního systému Symbian S60 a jeho implementací v současných SmartPhonech.

Konkrétně jsem se zaměřil na práci s běžnými typy dat a událostí v mobilních telefonech, jako jsou hovory, zprávy, kontakty, jejich skupiny, multimediální soubory a události při jejich přijetí, odesílání, respektive modifikaci, abych mohl navrhnout a implementovat aplikaci, která uživatelům umožní filtrovat SMS zprávy na základě předem vytvořených pravidel pro každé telefonní číslo, případně skupinu.

Vycházím z předpokladu, že v dnešní době moderních technologií a internetu je pro člověka důležité být stále v kontaktu. Život bez počítače, internetu a mobilního telefonu si málokdo dokáže představit. Technika se stále zdokonaluje a lidé se snaží, aby mělo co nejmenší zařízení co nejvíce funkcí, a díky tomu mohli mít vše potřebné stále po ruce. Mobilní telefon přestal být zařízením, které slouží jen k telefonním hovorům a k posílání SMS zpráv. Stalo se z něj multifunkční zařízení, které dokáže zaznamenávat hlas, přehrávat video, fotografovat, připojit se na internet a navíc si ještě můžeme ve volné chvíli zahrát nějakou hru či si naprogramovat vlastní software.

Pokud už náš telefon umí všechny tyto funkce, je s podivem, že nám stále není umožněno nastavit si rozdílné preference pro jednotlivé kontakty. Tato funkce by mohla být užitečná zejména v situacích, kdy si nepřejeme být rušeni zbytečnými zprávami, například z internetu, nebo zprávami, které obsahují určité slovo.

Rozhodl jsem se proto navrhnout aplikaci, která umožní uživatelům filtrovat SMS zprávy na základě telefonního čísla odesílatele nebo i jejich obsahu. Příchozí SMS zpráva může být díky nastavení ohlášena zvolenou melodií, neohlášena nebo přímo odstraněna z SMS inboxu. Činnost programu bude možno zaznamenávat do logu.

V první části práce je přiblížena struktura operačního systému. Jsou zde rozebrány důležité součásti systému, kterými jsou například jádro, file system, window server a další.

V části druhé se věnuji demonstrační aplikaci, která je hlavním předmětem této práce. Jedná se o filtr na zprávy SMS, pomocí kterého si uživatel může nastavit u jednotlivých kontaktů a jejich skupin druh upozornění, melodii, případně může filtr reagovat i na podezřelý obsah zprávy. Uvádím, jak jsem postupoval a jaké prostředky jsem používal. Nebude chybět ani ukázka, jak program funguje.

Kapitola 2

Symbian S60

2.1 Úvod

Symbian je v současné době nejrozšířenějším operačním systémem pro mobilní telefony. Je navržen pro telefony s ARM procesory. Zejména se jedná o SmartPhony – telefony s nadstandartním vybavením, mezi které patří například velký dotykový displej, wifi, úplná klávesnice a další množství funkcí.

V této kapitole se budu zabývat vznikem Symbianu, jeho historií a strukturou operačního systému, podrobněji rozeberu jeho důležité součásti.

2.2 Historie

Historie operačního systému Symbian začala už v roce 1980 ve firmě Psion, která se v té době zabývala výrobou kapesních počítačů.

Nejprve se jednalo o počítače bez operačního systému, s jedno či dvouřádkovým displejem, s možností naprogramovat si vlastní program v jazyce OPL.

S vývojem plnohodnotného operačního systému se začalo až v roce 1987. Šlo o multitaskingový systém EPOC, vydaný v 16 bitové (EPOC16 později známý jako SIBO) a 32 bitové (EPOC32) variantě. od roku 1991 byl EPOC součástí počítačů Psion Series 3 až 5, jež lze považovat za PDA. Mezi vybavení Psionů patřil například textový procesor, databáze, možnost připojení k internetu a další.

Od června 1998 je EPOC32 znám jako Symbian OS a je vyvíjen společností Symbian Ltd., která byla založena společnostmi Nokia, Ericsson, Motorola a Psion.

V roce 2004 prodal Psion svůj podíl v Symbian Ltd. a v červnu 2008 odkoupila Nokia zbylé podíly od ostatních firem, čímž se stala jediným vlastníkem společnosti.

Dne 24. června 2008 byla založena organizace Symbian Foundation, která si klade za cíl sjednotit co nejvíce firem a vytvořit kompletní platformu pro mobilní telefony. Tato platforma by navíc měla být od roku 2012 dostupná jako open source pod licencí Eclipse Public License.

V této části jsem vycházel ze zdrojů [7, 8, 9].

2.3 Struktura operačního systému

Operační systém je základní software, který řídí všechny funkce nejen počítače, ale právě i mobilního telefonu, na kterém je spuštěn. Je zodpovědný za správu připojeného hardwaru, za jeho řízení a integraci v systému, a také za správu softwaru, jeho instalaci, odebírání a samozřejmě i spouštění.

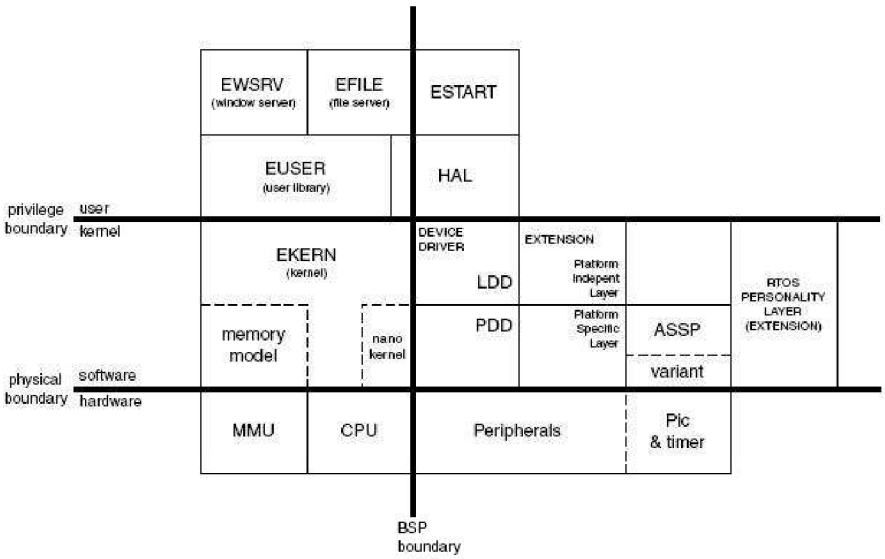
Operační systém je obvykle prvním programem, který je po bootu zařízení spuštěn. Jako první načte nezbytné ovladače a spustí základní aplikace nutné pro samotný běh systému. Operační systém poskytuje základní služby, kterými jsou například přístup do paměti, na disk, časování úkolů, práce s displejem a zajišťování správné komunikace mezi hardwarem a softwarem.

2.3.1 Kernel – EKA2

Jádro, neboli kernel, je základním stavebním kamenem každého operačního systému. Kernel Symbianu, EKA, vychází z jader EPOC16 a EPOC32. Zatímco jádra EPOC byla úzce svázána s platformou, na které měla běžet, u jader EKA tomu bylo jinak. Při jejich vývoji byl kladen důraz na nízkou závislost na platformě a na snadný budoucí vývoj dalších verzí.

V současné době se používá jádro EKA2. Toto jádro je real-timové a důraz je u něj kladen právě na modularitu.

Modularita zajišťuje snadnou spravovatelnost a nabízí lepší prostor pro vývojáře k upravování, případně k psaní vlastních komponent. Dá se říci, že modularita jádra EKA2 je na vyšší úrovni, než u velké části jiných operačních systémů. Funkcionalita je rozložena mezi samostatné servery. O práci s diskovými jednotkami a file systémy se stará file server, o uživatelské rozhraní, vstupy z klávesnice a displej se stará window server. O správu paměti se stará jádro. Vše je navrženo tak, aby bylo možné lehce modifikovat jakoukoliv část bez nutnosti zásahu do ostatních komponent systému. Celkové schéma operačního systému můžeme vidět na obrázku 2.1.



Obrázek 2.1: Schéma operačního systému Symbian [18]

EKA2 je jednouživatelský (single user) systém. To znamená, že neumožňuje přihlášení více uživatelů současně – tak, jak je tomu například u Linuxu nebo Mac OS X. EKA2 podporuje multi-tasking – přepínání CPU mezi několika vlákny (thready) tak, že má uživatel pocit, že běží v jednom okamžiku více aplikací. Abych byl přesný, tak se jedná o multitasking tzv. preemptivní, což znamená, že přidělování času jednotlivým vláknům probíhá v časových intervalech následujícím způsobem: jádro přeruší vykonávání právě probíhajícího vlákna, a vyhodnotí situaci, zjistí, které úlohy žádají o přidělení procesoru a jakou mají prioritu. Podle toho potom buď pokračuje v provádění přerušené vlákno, nebo vlákno jiné.

U real-timových operačních systémů (dále jen RTOS) není důležitá jen správnost výstupu, ale i doba, ve které je výsledek spočítán. RTOS se dělí na dvě skupiny právě podle typu reakce na překročení časového limitu (deadline). První skupinou jsou tzv. hard RTOS – pokud dojde k překročení časového limitu, je výsledek operace k ničemu a navíc je to považováno za chybu. U druhé skupiny – soft RTOS je překročení času tolerováno, ale musí se počítat s určitými následky v podobě například snížené kvality dané služby.

Vycházel jsem z těchto zdrojů: [18, 1].

2.3.2 Nanokernel

Nanokernel je součástí jádra a má na starosti ty nejprimitivnější operace. Jeho hlavním úkolem je plánování procesů a jejich synchronizace ve vláknech. Má také na starosti obsluhu přerušení. K synchronizaci jsou používány dva typy synchronizačních objektů: mutexy a semafory, o kterých se zmíním v části 2.3.3 *Synchronizační prostředky.*

Díky nanokernelu stačí výrobcům mobilních telefonů jen jedno CPU s jedním jádrem. Nanokernel totiž umožňuje provozovat RTOS s jeho GSM (Global System for Mobile communications) zásobníkem (GSM signal stack) a současně Symbian PIM (personal information management), což znamená pro výrobce výraznou úsporu prostředků. Navíc je zde možnost napsat si vlastní tzv. *personality layer*, neboli propojovací vrstvu, která poskytuje RTOS API (application programming interface) klientským programům. Díky této vrstvě je možno rozběhnout na Symbianu programy určené pro jiný RTOS.

Nanokernel používá svou vlastní knihovnu funkcí, přičemž nechává systémovou knihovnu `EUSER` dostupnou zbytku jádra a ovladačům.

Drtivou většinu času běží nanokernel v preemptivním módu. Výjimkou jsou některé velmi krátké části kódu obstarávající obsluhu vláken a synchronizaci. Děje se tak z důvodu zamezení přístupu některým vláknům do specifických částí kódu současně (změny stavu vláken) – a tím zachování deterministického chování.

Dynamická alokace paměti je součástí kernelu jako celku. Nanokernel neumí alokovat ani uvolňovat paměť. Proto se vždy musí spoléhat na to, že všechny paměťové prostředky jsou pro něj již alokovány.

Oddělené minijádro také usnadňuje emulaci. Emulátor Symbianu pro Windows se více podobá reálnému zařízení, což přináší větší spolehlivost než třeba emulátor s EKA1.

V této části jsem vycházel z [18].

2.3.3 Synchronizační prostředky

Mutex

Princip fungování mutexu uvedu na příkladě.

Mějme vlákno A a vlákno B. Tato vlákna vykonávají paralelně nějaký kód, ovšem v jednom okamžiku potřebují obě dvě sdílenou proměnnou X. Vlákno A její hodnotu zvýší o tři, vlákno B ji sníží o jedna a dále s ní pracuje. Tím nám vzniká nekonzistence, které je třeba zabránit. V tomto okamžiku přichází na scénu mutex, který kritickou část kódu, kde dochází k práci se sdílenými prostředky uzamkne, a pustí dovnitř pouze jedno vlákno. Druhé vlákno čeká, až se sekce uvolní.

Mutex je dvoustavový objekt – buď je *locked* (zamknut – vlákno je uvnitř) nebo *unlocked* (odemknut – v mutexu není žádné vlákno).

Semafor

Definice: „Semafor je synchronizační primitivum obsahující celočíselný čítač, který si lze představit například jako počítadlo volných prostředků. Poskytuje atomické operace *up* a *down*. Operace *down* sníží čítač o jedničku, v případě, že už je nulový (nedostává se prostředků), se proces zablokuje a přidá do fronty procesů čekajících na daný semafor. Operace *up* zkontroluje frontu, a v případě, že je neprázdná, vybere jeden proces čekající ve frontě a odblokuje jej (ten pak pokračuje za svou operací *down*); je-li fronta prázdná, zvýší hodnotu čítače o jedničku [5].“

Semafory se používají stejně jako mutexy pro synchronizaci vstupů do kritických sekcí. Ale na rozdíl od mutexu umožňují vstup do kritické sekce více vláknům současně. Počet takovýchto vláken je určen počítadlem volných prostředků (viz. definice).

Synchronizační prostředky jsou důležitou součástí pro práci s vlákny. Umožňují vláknům vstup do tzv. kritických sekcí.

„Za kritickou sekci považujeme tu část kódu vlákna, která operuje nad sdílenými daty a hrozí, že paralelně může jiné vlákno operovat nad stejnými daty. Důsledkem může být nekonzistence dat [4].“

V Symbianu se používají oba výše zmíněné synchronizační objekty.

2.3.4 Správa paměti, paměťový model

V této části se budu zabývat správou paměti a paměťovým modelem.

Symbian je 32bitový operační systém. To znamená, že pro uložení paměťových adres musí stačit 32bitový registr. Z tohoto plyne omezení adresovatelného paměťového prostoru na 4 GB.

V současné době se v telefonech vyskytují maximálně operační paměti s kapacitou 128 MB až 256 MB, takže je adresovatelný prostor dostatečně velký.

Jak jsem se již zmínil výše, správu paměti má na starosti kernel. Jeho hlavními úkoly ve spojitosti se správou paměti jsou následující:

- správa fyzických zdrojů paměti: RAM, cache a MMU,
- alokace fyzické a virtuální paměti,
- správa adresového prostoru každého procesu,

- izolace procesů navzájem,
- ochrana paměti.

Mimo to, vývojáři stanovili ještě další limity, které by měl být systém schopen dodržet.

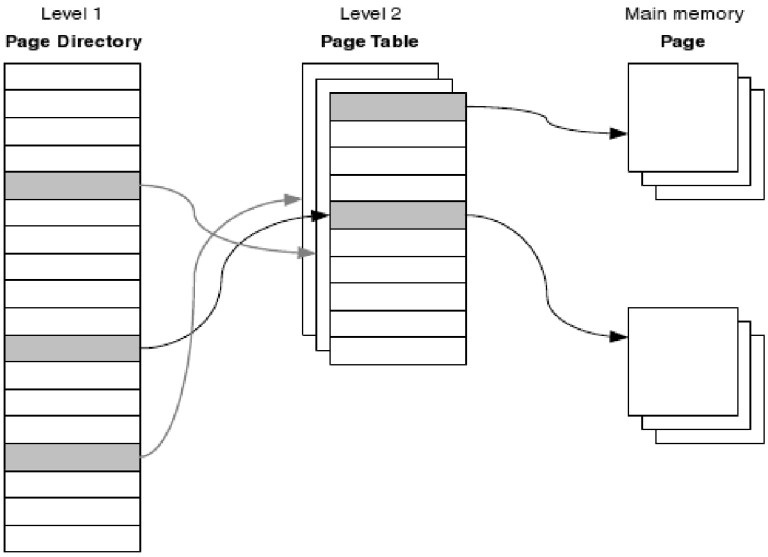
- Počet procesů by neměl být omezen paměťovým modelem, ale volnými zdroji v systému.
- Každý proces by měl mít k dispozici adresový prostor o velikosti 1 GB až 2 GB.
- Objem spustitelného kódu, který může být procesem použit, by měl být omezen pouze dostupnou pamětí RAM či ROM.

Vývojáři věděli, že operační systém bude provozován na různých, rychle se vyvíjejících zařízeních. Proto bylo nutno zajistit způsob, který by dovoloval snadnou migraci systému na nový hardware s co nejmenšími zásahy do systému. Proto došlo k jistému odloučení kódu pro správu paměti od jádra. Sice je jeho součástí, ovšem takovým způsobem, aby bylo možné se přizpůsobit novému hardware s odlišnou MMU a paměťovou architekturou.

MMU

Jednotka MMU (memory management unit) je součástí procesoru. Dělá prostředníka mezi fyzickou pamětí a pamětí virtuální. Jejím úkolem je překlad virtuálních adres, které chápe software, na adresy fyzické, které pochopí hardware. Tento překlad se děje při každém přístupu do paměti.

MMU rozděluje fyzickou paměť na bloky nazývané stránky. Velikost stránek je obyčejně 4 KB, ale mohou se vyskytnout i stránky o velikosti 64 KB či 1 MB. Stránky jsou uspořádány v tabulce stránek (page table) a jednotlivé tabulky zas v adresáři stránek (page directory). Celé schéma je naznačeno na obrázku 2.2.



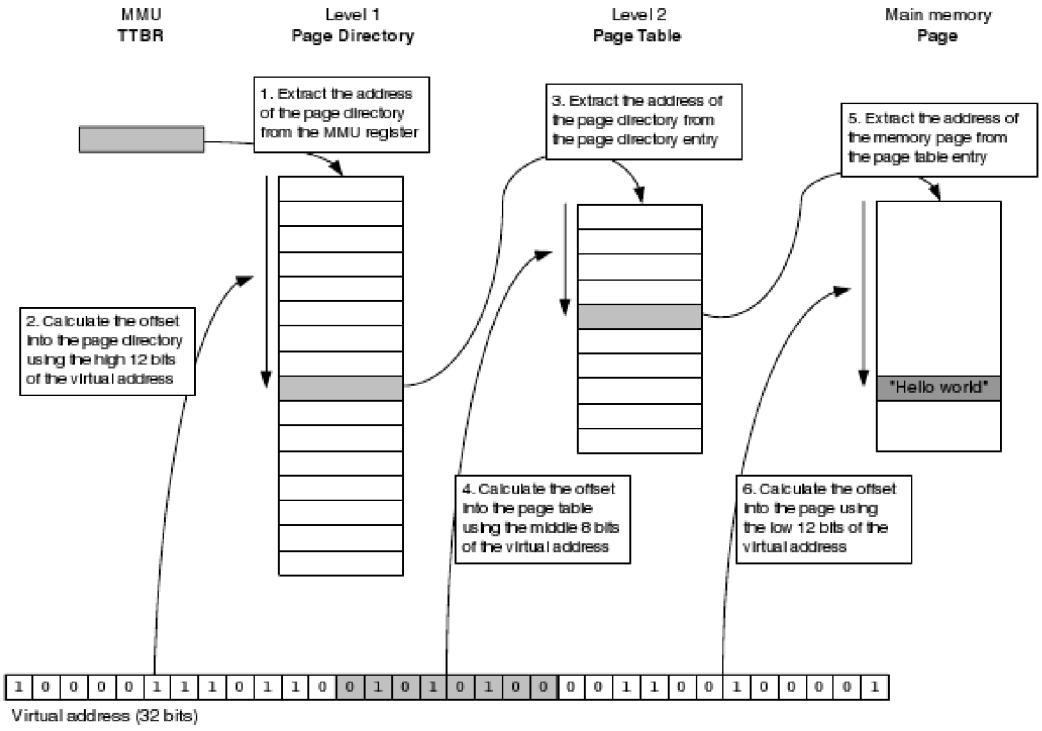
Obrázek 2.2: Struktura adresáře stránek [18]

Tabulka 2.1: Struktura virtuální adresy

Bity	**31 → 20**	**19 → 12**	**11 → 0**
Význam bitů	Horních 12 bitů reprezentuje adresář stránek	Prostředních 8 bitů reprezentuje tabulku stránek	Dolních 12 bitů určuje offset stránky

Struktura virtuální adresy je patrná z tabulky 2.1.

Při každém přístupu do paměti se provede prohledání tabulek stránek a následný překlad virtuální adresy na adresu fyzickou. Nejdříve se najde v adresáři stránek tabulka stránek a poté se provede vyhledání příslušné stránky. Algoritmus překladu adres je popsán na obrázku 2.3.



Obrázek 2.3: Algoritmus překladu virtuálních adres [18]

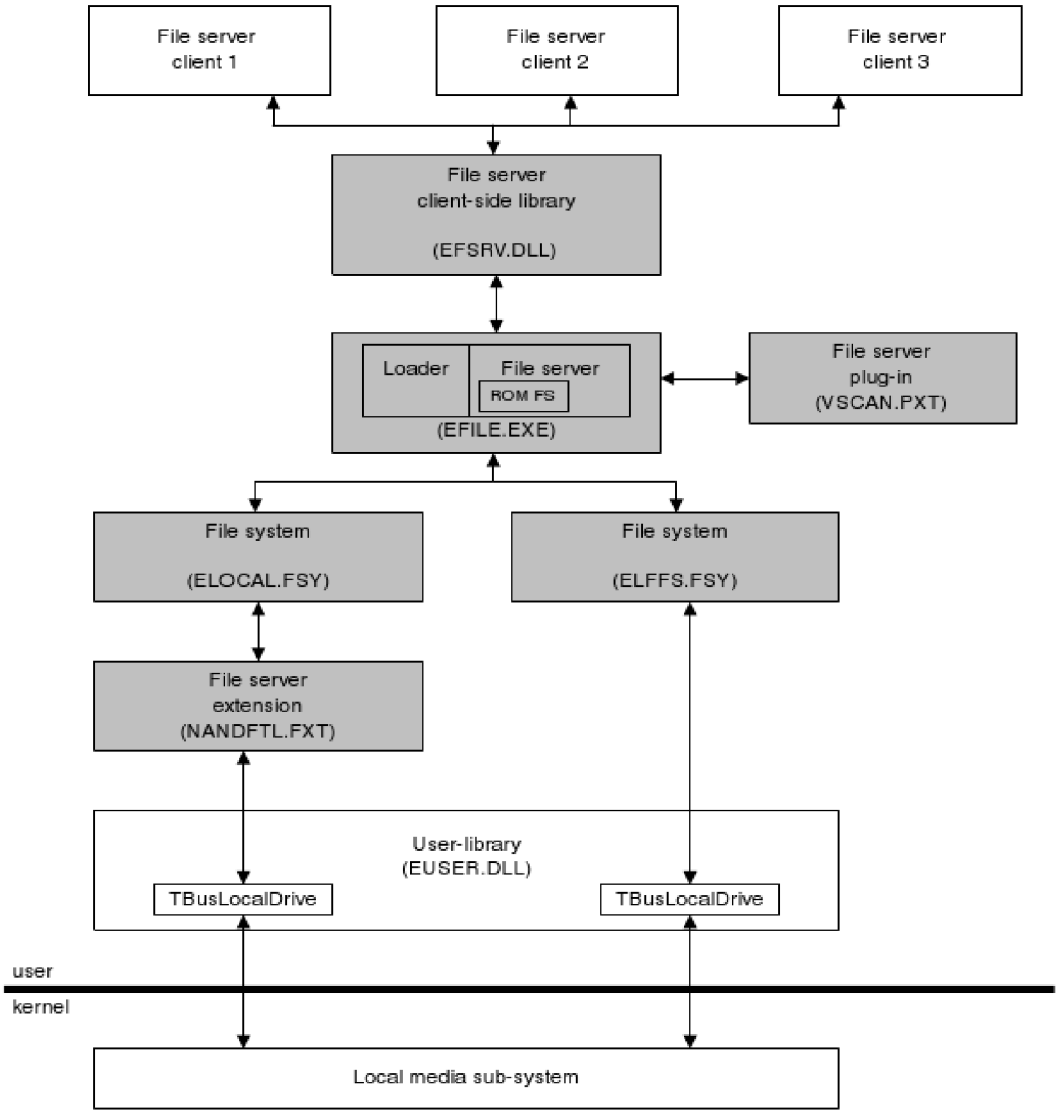
Pro urychlení procesu vyhledávání si MMU uchovává několik posledních výsledků hledání v tzv. Translation Look-aside Bufferu (dále jen TLB). Pokud se výsledek nenachází v TLB provede se nové hledání a výsledek je uložen do TLB. Samozřejmě, pokud dojde ke změně tabulky stránek, musí být TLB smazáno.

2.3.5 File server

File server, někdy nazývaný F32, má na starosti správu diskových jednotek. Umožňuje přistupovat k souborům, adresářům a jednotlivým oddílům na disku a také zajišťuje připojování (mountování) výměnných disků.

Součástí file serveru je také tzv. loader, který se stará o načítání dynamicky linkovaných knihoven (soubor `DLL`) a spouštění aplikací (soubor `EXE`). Loader samotný přiblížím v části 2.3.6 *Loader*.

File server je jako velká část věcí v Symbianu založen na klient-server architektuře – od klientů získává požadavky, které následně zpracovává. Je multivláknový, což zvyšuje jeho efektivitu. Klienti k přístupu k file serveru používají API knihovny `EFSRV.DLL`.



Obrázek 2.4: Schéma file serveru F32 [18]

Struktura file serveru

Samotný file server sestává z částí vyznačených šedě na obrázku 2.4.

Jak je z obrázku 2.4 patrné, centrální komponentou je `EFILE.EXE` obsahující jádro file serveru, loader a navíc plugin pro souborový systém ROM.

Proč je tento plugin už zabudovaný ve file serveru? Odpověď zní následovně. Při bootování systému je hned po zavedení jádra nutné spustit další aplikace – zejména ty základní, které jsou uloženy v ROM. Je použit speciální mechanismus, který v době, kdy není k dispozici ani loader ani ROM file system spustí file server. v tomto okamžiku je nainstalován ROM file system a následně loader, který spustí všechny zbývající aplikace. Tento způsob by nebyl možný bez vestavěného pluginu pro ROM file system v `EFILE.EXE` .

Další součástí file serveru je už výše zmíněná knihovna `EFSRV.DLL`, která poskytuje klientům API k file serveru.

Pluginy jsou další součástí. Jsou to oddělené komponenty s koncovkou `.PXT`, které se spojí s file serverem a jsou schopny zpracovávat požadavky od klientů. Například se může jednat o pluginy, které mají na starosti šifrování a dešifrování souborů, skenování souborů, zda neobsahují virus a podobně. Výhodou je, že lze každý plugin nastavit tak, aby se spouštěl jen v určitých situacích nebo jen pro určité adresáře či jednotky.

Poslední částí file serveru jsou soubory reprezentující systémy souborů a jejich rozšíření. Samotný file server se nestará přímo o strukturu systému souborů na jednotlivých médiích. Pro každý souborový systém existuje jeho reprezentace v podobě jednoho souboru. Při připojení diskového zařízení se daný soubor načte a připojí k file serveru. File server mu poté předkládá své příkazy a on je převede do formy vhodné pro daný souborový systém.

Podporované systémy souborů

- **ROM** – Read Only Memory – speciální file system pro uložení kódu přímo na médiu, ze kterého se bude spouštět (NOR flash), navrhnut tak, aby se dosáhlo co nejefektivnějšího čtení.
- **LFFS** – Log Flash File System – používá se ve flash pamětích složených z NOR hradel pro uložení uživatelských dat.
- **FAT** – File Allocation Table – používá se ve flash pamětích složených z NAND hradel, pro uživatelská data stejně jako LFFS, navíc se s ním ještě můžeme setkat v RAM jednotkách a na výměnných médiích.
- **ROFS** – Read Only File System – stejný jako ROM, ovšem s tím rozdílem, že kód uložený na médiu s ROFS musí být před spuštěním zkopírován do RAM, vyskytuje se na NAND flash.
- **Složený (composite) FS** – není file systémem v pravém slova smyslu. Slouží jako vrstva propojující ROM a ROFS file system. Toto propojení je zejména nutné pro zpřístupnění potřebných dat na přístroji, který používá NAND hradlo pro uložení kódu, který je potřeba v čase bootování.

Symbian označuje jednotky velkým písmenem a dvojtečkou (DOS notace). Prostor je tu pro dvacet šest jednotek (A: až Z:). Jednotka s ROM obrazem je vždy označena jako Z:, hlavní uživatelská jednotka potom C: a výměnná úložiště jsou obvykle značeny písmeny D: či E:.

File server podporuje dlouhé názvy souborů. Maximální délka je 256 16bitových znaků.

Celý název souboru se skládá ze čtyř komponent:

- **písmeno jednotky** – písmeno a dvojtečka,
- **cesta** – seznam adresářů, započatý, ukončený a oddělený zpětnými lomítky,
- **jméno souboru** – vlastní jméno souboru, může být ukončeno tečkou, za kterou následuje přípona,
- **přípona** – skládá se ze znaků následujících za poslední tečkou.

V části 2.3.5 *File server* jsem vycházel z knihy [18].

2.3.6 Loader

Loader je součástí file serveru. Je to druhý server, který běží v samostatném vlákně a má na starosti spouštění souborů a načítání potřebných knihoven. Opět se jedná o část, která je postavena na architektuře klient-server.

Spustitelné soubory musí být ve formátu E32.

Každý takovýto soubor se skládá z devíti částí:

- **hlavička**,
- **část s kódem – .text** – zde je uchováván spustitelný kód,
- **část s konstantními daty – .rdata**,
- **tabulka importovaných adres (IAT)** – obsahuje záznam pro každou importovanou funkci,
- **export directory – .edata** – tabulka, kde se uchovávají adresy všech vyexportovaných funkcí,
- **část s inicializovanými daty** – tato část obsahuje data, která se zkopírují při spuštění souboru do RAM,
- **část s importovanými daty** – .idata – zde jsou uložena data pro každou funkci, která je importována, tato část je používána pro určení dynamických knihoven, které je nutno načíst,
- **relokační část pro kód** – zde jsou uloženy všechny relokace, které se mají aplikovat v kódové části,
- **relokační část pro data** – zde jsou uloženy všechny relokace, které se mají aplikovat v datové části.

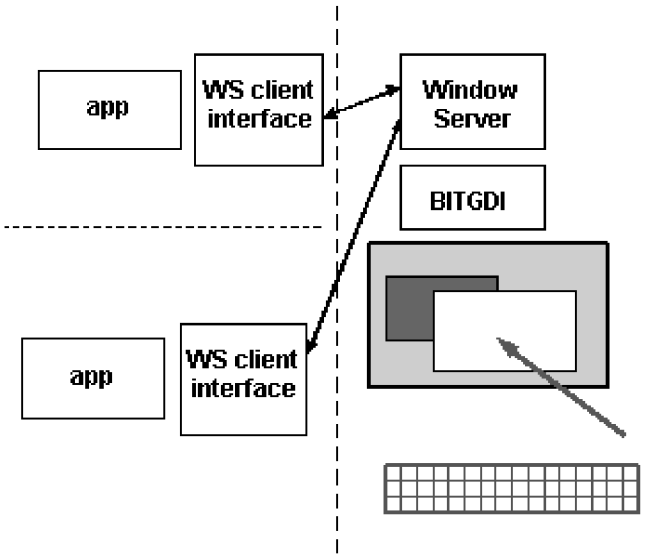
Struktura spustitelného souboru uloženého na disku je odlišná od struktury, která vznikne po načtení souboru loaderem. Po načtení loaderem totiž vzniknou sekce pouze dvě. První z nich je kódová část, kde je obsažen vlastní kód, konstantní data, IAT a export directory. V druhé, datové, části jsou uložena jak inicializovaná, tak neinicializovaná data.

Při psaní této části jsem vycházel z [18].

2.3.7 Window server

Window server (či WSERV) poskytuje aplikacím rozhraní pro komunikaci s uživatelem. Obstarává propojení mezi uživatelskými vstupy, ať už z klávesnice, joysticku nebo touchscreenu, a displejem.

WSERV slouží kernelu jako obsluha událostí. Kdykoliv se v kernelu objeví nová událost související s uživatelským rozhraním, je předána window serveru, který ji zpracuje a přepošle příslušnému klientovi. Druhy událostí předávaných mezi kernelem a window serverem se mohou lišit od událostí, které předává window serverem klientům. WSERV také může některé události „zahodit“, případně generovat události vlastní.



Obrázek 2.5: Schéma window serveru [10]

Spuštění window serveru probíhá během startu systému a jeho priorita je velmi vysoká (vyšší prioritu má jen samotný kernel). Díky tomu je zajištěno přednostně zpracování událostí, jako například zmáčknutí tlačítka, požadavku na překreslení displeje a dalších.

Každá klientská aplikace komunikuje se serverem samostatně ve svém vlastním vláknu – WSERV zajišťuje, aby se aplikace navzájem neovlivňovaly. Vykreslování se provádí pouze do viditelných částí oken a události jsou oknem přijímány pouze v případě, že s ním přímo souvisí a nebo v případě, že je okno v popředí (má focus). Události je možné zasílat i oknům, která sice nejsou v popředí (nemají focus), ale řekla si o zasílání určitých událostí. Aplikace obvykle ignorují většinu událostí patřící jiným oknům a o velké části z nich se ani nedoví.

V této části práce jsem vycházel ze zdrojů [10, 18].

Kapitola 3

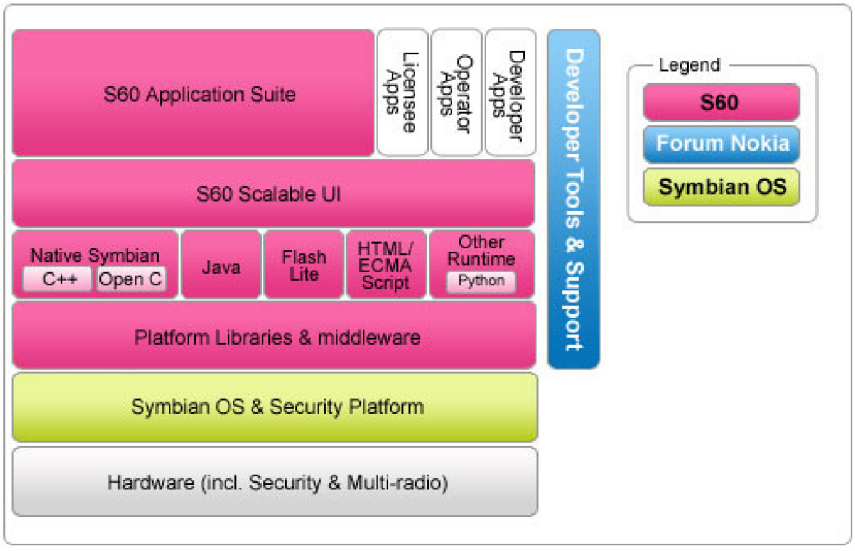
Demonstrační aplikace

3.1 Zadání

Jako demonstrační aplikaci jsem měl podle zadání vytvořit jednoduchý SMS filtr. Aplikace by měla zvládat reagovat na příchozí zprávy SMS a podle uloženého nastavení provádět pro daný kontakt akci, kterou si uživatel přednastavil. Může to být přehrání vyzváněcího tónu nebo naopak jeho potlačení. Uživatel může také chtít zprávu přímo odstranit z inboxu bez jakéhokoliv upozornění anebo prohledat obsah zprávy, a pokud zpráva obsahuje daný výraz, tak provést nějakou z výše zmíněných akcí. Aplikace by měla být uživatelsky přívětivá a měla by být schopna běžet v pozadí.

3.2 Vývojové prostředky

Před vlastním návrhem aplikace jsem se rozhodoval mezi několika dostupnými programovacími jazyky pro Symbian. Možnosti, které jsem zvažoval jsou znázorněny na obrázku 3.1. v následující části uvádím jejich krátké představení.



Obrázek 3.1: Schéma platformy Symbian OS [11]

3.2.1 Open C a C++

Open C a C++ jsou programovacími jazyky, které vznikly upravením klasického C/C++ pro potřeby Symbianu. Jedná se o jakýsi souhrn knihoven pro platformu S60. Knihovny Open C jsou založeny na P.I.P.S. (P.I.P.S. is POSIX on Symbian OS) a dalších pěti open source knihovnách jazyka C.[14]

3.2.2 Java

Java je interpretovaný, objektově orientovaný programovací jazyk, který není závislý na architektuře stroje, na kterém běží. Je to jeden z nejpoužívanějších jazyků na světě a vyvinula ho v roce 1995 firma Sun Microsystems.

Díky tomu, že je Java multiplatformní jazyk, je používána v mnoha různých systémech od čipových karet přes mobilní telefony až po osobní počítače a distribuované systémy pracující na řadě spolupracujících počítačů po celém světě.

Od roku 2007 je Java vyvíjena jako open source.

V prostředí Symbianu se setkáváme s Javou ME (Micro Edition), která je určena právě pro zařízení s omezenými systémovými prostředky, jimiž mobilní telefony v současné době bezesporu jsou.

Nevýhodou je, že aplikace napsané v Java ME se načítají pomaleji než v C++. Je to samozřejmě kvůli tomu, že tento jazyk běží ve virtuálním stroji. Navíc má Java omezený přístup k funkcionalitě telefonu (například se nedá dostat do inboxu a manipulovat s SMS zprávami, nedá se ani posílat klasické SMS zprávy), což může být u některých aplikací nepřekonatelný problém.

Pro mou práci to znamenalo přechod k Pythonu, protože práce s SMS zprávami byla pro mě elementární záležitostí. Nicméně se Java výborně hodí pro psaní her a jednodušších aplikací, kde není třeba pracovat s SMS inboxem a kde nevadí ani její nižší rychlost [13].

Pro používání Javy v mobilních telefonech je důležité, aby mobilní telefon podporoval tyto standardy:

CLDC

Connected Limited Device Configuration (CLDC) je specifikace, která definuje požadavky pro běh základních aplikací, virtuálního stroje a API v zařízeních s omezenými systémovými prostředky. CLDC je postaveno na sandboxu, který zajišťuje ochranu před špatně napsanými aplikacemi.

Základními požadavky pro splnění této specifikace (CLDC 1.0) jsou:

- alespoň 16bitový procesor,
- alespoň 128 KB ROM paměti pro virtuální stroj a CLDC knihovny,
- alespoň 32 KB RAM paměti pro vlastní aplikace,
- operační systém musí umět pracovat s virtuálním strojem,
- operační systém musí být schopen instalovat, spouštět a odebírat aplikace.

V kombinaci s MIDP poskytuje CLDC platformu pro vývoj Java aplikací pro zařízení s omezenou pamětí, rychlostí procesoru a grafických prostředků [2, 15].

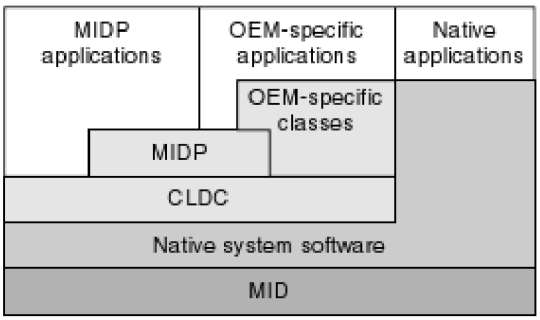
MIDP

Mobile Information Device Profile (MIDP) je rozšířením CLDC. Jedná se o vrstvu naházející se „nad“ CLDC (viz. 3.2) a definující charakteristické vlastnosti aplikací:

- podpora uživatelského rozhraní,
- podpora HTTP,
- podpora úložiště dat.

MIDP navíc poskytuje různé třídy jazyka Java (výjimky, chyby, třídy pro práci se sítí).

Všechny aplikace splňující podmínky dané CLDC a MIDP se nazývají MIDlety. MIDlety mohou být spouštěny na jakémkoliv zařízení podporující MIDP [2, 16].



Obrázek 3.2: Architektura MIDletu [2]

3.2.3 Python

O Pythonu

Python je interpretovaný, objektově orientovaný programovací jazyk. Často je zařazován do skupiny skriptovacích jazyků, k Perlu a Ruby. Autorem je nizozemský programátor Guido van Rossum, který jej vytvořil v roce 1990.

Python klade důraz na jednoduchost, bývá dokonce často označován jako jeden z jazyků, které je možno doporučit programátorským začátečníkům pro jeho snadnou pochopitelnost a názornost. Vychází z jazyka ABC, který byl určen právě začátečníkům.

V současnosti je Python dostupný pro většinu běžných platforem – Unix, Mac OS, Windows, Palm, mobilní telefony Nokia a další, a je vyvíjen jako open source projekt. V základní instalaci uživatel obdrží množství modulů a další není problém dohledat třeba na internetu.

Vycházel jsem z [17].

Instalace Pythonu pro S60

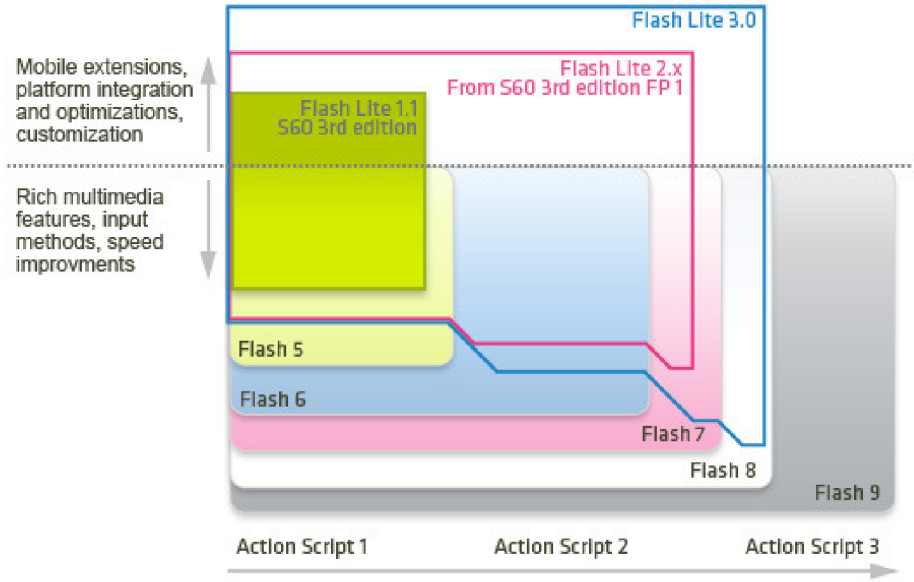
Abychom mohli Python používat na platformě S60 je nutné jej nejprve nainstalovat, protože není její nativní součástí. Vše potřebné najdeme na stránkách [3]. V době vzniku této práce byla aktuální verze 1.9.2.

Pro instalaci budeme potřebovat samotný interpret jazyka s moduly a Python script shell. V mém případě se jednalo o soubory Python_1.9.2_3rdEd.sis (interpret) a PythonScriptShell_1.9.2_3rdEd.sis (shell). Používám telefon Nokia E70 s CLDC 1.0 a MIDP 2.0. Modely mohou mít rozdílnou konfiguraci, proto je nutné vybrat správný soubor dle typu konkrétního mobilního telefonu a verze operačního systému.

Pokud jsou staženy oba soubory, stačí je už jen nainstalovat běžným způsobem. Nejprve je nutno instalovat interpret a až poté shell.

3.2.4 Flash Lite

Mobilní telefony se Symbianem standardně podporují Flash. Flash Lite, jak se mobilní Flash jmenuje, je implementací klasického Flashe tak, jak ho známe z internetového prohlížeče (viz. obrázek 3.3). Je navržen s ohledem na omezené systémové prostředky mobilního telefonu, ale zato je rozšířen o možnost posílat SMS, MMS nebo vytočit telefonní číslo.



Obrázek 3.3: Návaznost Flash Lite a Flash [12]

3.2.5 Qt

Qt zná většina z nás spíše z unixových operačních systémů. v současné době byla oznámena plánovaná podpora systému Symbian.

Jedná se o framework napsaný v C++ umožňující vývoj aplikací s uživatelskými rozhraními. Implementace Qt existuje i pro Python, Perl, Pascal, Ruby, Javu a další. Výhodou Qt je možnost navrhnout aplikace a poté je používat na různých platformách bez nutnosti zásahu do zdrojového kódu. Qt je multiplatformové, v současnosti podporuje platformy Windows, Mac, Linux, Windows CE, Windows Mobile a embedded Linux.

S budoucí podporou Symbianu se uživatelům i vývojářům otevřou nové možnosti vývoje aplikací.

Bohužel jsem informaci o podpře Qt zachytil až v době, kdy jsem měl demonstrační aplikaci téměř hotovou v jazyku Python, navíc je Qt pro Symbian stále ve fázi vývoje a byly uvolněny pouze testovací verze.

3.3 Návrh řešení

Zadání mi ukládalo vytvořit aplikaci, která bude demonstrovat funkčnost Symbianu při práci s běžnými typy dat a událostí, se zprávami, kontakty, jejich skupinami, multimediálními soubory a událostmi spojenými s jejich přijetím, případně odesíláním.

Rozhodl jsem se vytvořit aplikaci pro filtrování zpráv SMS.

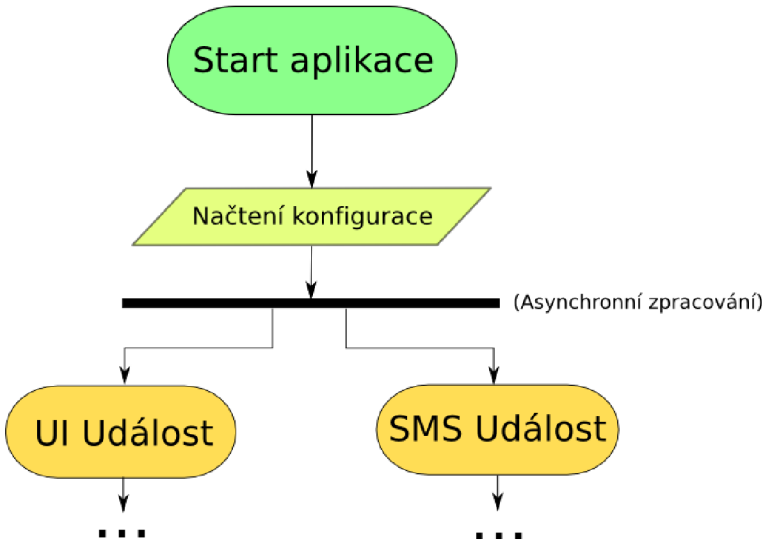
Pro tento účel bude nutno vytvořit konfigurační soubor s vhodnou strukturou pro uchovávání nastavení pravidel. Tento `config` bude mít strukturu konfiguračního `.INI` souboru – tyto soubory jsou známy například ze systému Windows. Příklad toho, jak může `config` vypadat je znázorněn na obrázku 3.1 v části 3.4.3 *Struktura konfiguračního souboru.*

Dále bude nutno vytvořit vhodné datové struktury pro uchovávání těchto pravidel v paměti a práci s nimi. Také nesmím zapomenout na metody umožňující práci s konfiguračním souborem – ukládání a načítání pravidel.

Potřebné budou metody pro přidávání nových pravidel, zobrazování a mazání starých pravidel a pro věci s tím spojené jako například zobrazování seznamu kontaktů, výběr melodie a další.

Na závěr vytvořím nejdůležitější část, a sice metody, které budou reagovat na příchozí zprávu, zkontrolují pravidla a provedou příslušnou akci.

Schéma toho, jak bude aplikace fungovat je znázorněno na obrázcích 3.4, 3.5 a 3.6.



Obrázek 3.4: Část schematu demonstrační aplikace, pokračování viz. obrázky 3.5 a 3.6

Pro vlastní implementaci jsem si po delší rozvaze vybral programovací jazyk Python pro S60 (PyS60). V úvahu připadal ještě jazyk Java ME, který se však ukázal jako nevhodný, protože kvůli sandboxu není programátorovi dovoleno přistupovat do SMS inboxu.

Při studiu jsem se setkal s jazyky C a C++, Javou a několika dalšími. I přesto, že jsem s Pythonem nikdy dříve nepracoval, rozhodl jsem se pro něj pro jeho údajnou jednoduchost, dobrou provázanost s platformou S60 a také jsem věděl, že se nebudu muset starat o alokaci paměti a řešit problémy s tím spojené, protože se Python o tyto věci postará sám.

3.4 Implementace

Při implementaci jsem postupoval následovně. Nejprve jsem navrhl strukturu konfiguračního souboru a vytvořil metody pro jeho čtení a zápis.

Ve chvíli, kdy toto spolehlivě fungovalo, započal jsem s vývojem metod pro přidávání nových pravidel.

Nejprve jsem navrhl a vytvořil metody pro filtrování kontaktů, s metodami pro skupiny jsem se vypořádal až nakonec. Jakmile jsem měl hotové přidávání pravidel, přišlo na řadu zpracování příchozích zpráv a procházení pravidel.

Závěrem jsem přidal zobrazování a mazání pravidel a především logování do souboru.

3.4.1 Třída `SMSfilter`

Třída `SMSfilter` je hlavní třída, která obstarává chod aplikace.

Obsahuje metody pro práci s konfiguračním souborem, pro přidávání a odebírání pravidel a hlavně pro vyhodnocování příchozích SMS zpráv a provádění příslušných akcí. Zajišťuje mimo jiné také grafickou stránku aplikace.

Nyní následuje popis jednotlivých metod.

- `__init__(self)`

  Speciální metoda Pythonu, která zajišťuje inicializaci programu.

  Provádím zde nastavení jména aplikace, nastavení funkce levému softkey tlačítku – zobrazení menu, nastavení funkce pravému softkey tlačítku – `exit_handler()` (viz. níže).

  Dále zde inicializuji komponentu pro logování, a načítám obsah souboru s nastavením pravidel pro filtrování zpráv do příslušných polí pomocí metody `readConfig()`.

- `run(self)`

  Výchozí metoda, která se volá po inicializaci.

  Provede se zde napojení (nabindování) metody `newSmsHandler()` na inbox. Díky tomu budou příchozí zprávy zpracovány právě touto metodou.

- `newSmsHandler(self, msg)`

  Metoda, která má na starosti zpracování příchozí SMS zprávy.

  Nejdříve získá telefonní číslo odesílatele SMS, potom je tělo zprávy převedeno na malá písmena (prohledávání textu příchozí zprávy není case sensitive – slova 'Spam' a 'SPAM' jsou ekvivalentní). Následně se provede zalogování příchozí zprávy, zkontrolují se pravidla a provede se daná akce.

- `applyBasicRules(self, number)`

  Tato metoda se stará o kontrolu a aplikaci pravidel pro samotná telefonní čísla (`basic`).

  Jejím jediným úkolem je projít pole `basicRules`, a pokud zde existuje záznam pro telefonní číslo `number`, provést příslušnou akci a vrátit `True / False`.

- `applyGroupRules(self, number)`

  Obdoba metody `applyBasicRules()` s tím rozdílem, že prochází pravidla pro skupiny (`group`).

- `applyDefaultRule(self)`

  Metoda, která se volá v případě, že pro číslo není nastavena žádná speciální akce – metody `applyBasicRules()` i `applyGroupRules()` vrátily `False`.

- `addRule(self)`

  Metoda, pomocí které se přidává nové pravidlo.

  Nejprve se zobrazí vyskakovací menu s nabídkou typu pravidla, které se bude přidávat. Může se jednat o pravidlo pro jediné telefonní číslo (`basic`), pro skupinu (`group`) nebo o změnu nastavení základního pravidla (`default`). V dalším kroku se zavolají metody pro výběr telefonního čísla (`pickContact()`) či skupiny (`pickGroup()`), akce, která se provede (`pickAction()`), melodie (`pickMelody()`) a případně, pokud uživatel vybral jako akci filtrování obsahu zprávy, se zobrazí okno pro zadání textu, který se bude ve zprávě vyhledávat a ještě se vybere požadovaná akce (`pickSmsAction()`).

  Nakonec se pravidlo přidá do příslušného pole pravidel (`basicRules`, `groupRules`, `defaultRule`) a zavolá se metoda `writeConfig()`, která pravidlo uloží do konfiguračního souboru.

- `pickAction(self)`

  Tato metoda slouží k nastavení akce, která se provede při přijetí zprávy. Jsou čtyři možnosti:

  1. `ring` – přehraje se melodie,
  2. `mute` – nepřehraje se melodie – bude ticho,
  3. `check_sms` – zkontroluje se obsah zprávy,
  4. `block` – zpráva se odstraní bez jakéhokoliv signálu.

- `pickContact(self)`

  Tato metoda je určena k výběru telefonního čísla.

  Nejprve se projde seznam kontaktů a vyberou se mobilní čísla (`mobile_number`). V případě, že není u kontaktu uvedeno mobilní číslo, vybere se číslo na pevnou linku (`phone_number`).

  Čísla se ukládají do asociativního pole, kde jako klíč slouží jméno kontaktu. Vznikne seznam, který je seřazen a zobrazen uživateli jako seznam kontaktů. Uživatel si vybere podle přezdívky telefonní číslo a to je metodou vráceno.

  Práci s kontakty jsem musel řešit takto, protože v PyS60 neexistuje modul, který by otevření kontaktů umožňoval přímo.

- `pickGroup(self)`

  Tato metoda je určena k výběru skupiny.

  Pracuje podobně jako (`pickContact()`), jen s tím rozdílem, že neprochází celý seznam kontaktů, ale pouze seznam skupin.

- `pickMelody(self)`

  Metoda, která slouží k získání cesty k melodii.

  Zde jsem použil třídu `FileSel`, převzatou z internetových stránek[6]. Tato třída poskytuje file browser, pomocí kterého si uživatel vybere melodii. Metoda vrací cestu k melodii.

- `pickSmsAction(self)`

  Metoda sloužící k nastavení akce, která bude následovat po tom, co bylo v textu SMS zprávy nalezeno hledané slovo(viz. `pickAction(self)`).

- `getMobile(self, title)`

  Metoda, která na základě jména kontaktu `title` prohledá seznam kontaktů a vrátí příslušné telefonní číslo.

- `getNickname(self, number)`

  Metoda, která na základě telefonního čísla `number` prohledá seznam kontaktů a vrátí příslušné jméno kontaktu.

- `getGroupName(self, gID)`

  Metoda, která na základě ID skupiny vrátí její název, případně `None`, pokud kontakt s tímto ID není skupinou.

- `removeRule(self, ind)`

  Metoda odstraní pravidlo na dané pozici `ind` v poli `basicRules` či `groupRules`.

- `showRules(self)`

  Metoda zobrazí seznam pravidel kromě základního pravidla (`default`). Uživatel může pomocí zvolení některého pravidla toto pravidlo odstranit. V případě, že k výběru skutečně dojde, je zavolána metoda `handle_listbox`, která se postará o smazání pravidla a uložení konfiguračního souboru.

- `readConfig(self)`

  Tato metoda se volá při inicializaci programu a jejím úkolem je načtení konfiguračního souboru umístěného v `c:\Data\SMSfilterConfig.txt`.

  Pro snadnější manipulaci metoda načte pravidla a jejich počet do příslušných polí a proměnných:

  - `ruleCount` – celkový počet všech pravidel (`basic` + `group`),
  - `basicRuleCount` – počet pravidel pro telefonní čísla (`basic`),
  - `groupRuleCount` – počet pravidel pro skupiny (`group`),

  - `basicRules` – pole s pravidly pro telefonní čísla,
  - `groupRules` – pole s pravidly pro skupiny,
  - `defaultRule` – asociativní pole se základním pravidlem.

- `writeConfig(self)`

  Metoda, která slouží k uložení konfigurace do konfiguračního souboru.

  Prochází se postupně výše zmíněná pole s pravidly, a ta se ukládají do konfiguračního souboru. O struktuře samotného konfiguračního souboru se více zmíním v části 3.4.3 *Struktura konfiguračního souboru.*

- `back(self)`

  Tato metoda slouží jako obsluha pravého softkey tlačítka.

  Používá se v případě, že je otevřeno menu a obstarává návrat do předchozí nabídky, resp. do hlavního menu.

- `handle_listbox(self)`

  Obslužná metoda, která obstarává smazání zvoleného pravidla. Volá se v metodě `showRules()` a sama volá metodu `removeRule()`, jíž předává index mazaného pravidla.

- `exit_handler(self)`

  Poslední z obsluhovacích metod. Jejím úkolem je ukončit program. Volá se při zmáčknutí červeného tlačítka, případně pokud je v hlavním menu zvolena možnost *Exit*.

- `loggerInit(self)`

  V programu je pro logování použit modul `logging`. Metoda `loggerInit()` slouží k inicializaci loggeru. Nastaví se zde formát záznamu, maximální velikost logu a další.

- `logSettings(self)`

  Metoda, která má za úkol zobrazit menu pro zapnutí/vypnutí logování a nastavení uložit do konfiguračního souboru.

3.4.2 Třída `FileSel`

Tato třída slouží k procházení souborů v telefonu a převzal jsem ji z internetových stránek [6]. Pomocí masky lze určit typ souborů, který půjde zvolit. Třídu jsem částečně upravil, tak, aby vracela cestu k souboru.

- `__init__(self)`

  Zde se nastaví výchozí adresář, maska a zavolá se metoda `fill_items()`.

- `fill_items(self)`

  Úkolem této metody je načíst seznam adresářů a souborů v aktuálním adresáři.

- `run(self)`

  Metoda, jejíž hlavní částí je nekonečná smyčka, ve které se zobrazí obsah aktuálního adresáře a čeká se na akce uživatele. Pokud uživatel vybere adresář, zobrazí se jeho obsah. Pokud je vybrán soubor, vrátí se celá cesta k souboru.

3.4.3 Struktura konfiguračního souboru

Konfigurační soubor slouží k uchovávání nastavených pravidel pro telefonní čísla.

V Pythonu existuje modul `ConfigParser`, který dovede s konfiguračními soubory pracovat.

Samotná struktura se shoduje s konfiguračními `.INI` soubory známými např. z Windows.

Tabulka 3.1: Struktura konfiguračního souboru

```
[DEFAULT]
action = ring
melody = C:\Data\Sounds\Simple\one.mp3
sms_action = None
sms_text = None
[0]
sms_text = spam
action = check_sms
melody = C:\Data\Sounds\Simple\two.mp3
type = basic
id = +420776580067
sms_action = ring
[1]
sms_text = None
action = ring
melody = C:\Data\Sounds\Simple\three.mp3
type = group
id = 458
sms_action = None
[config]
rulecount = 2
enable_logging = True
```

Jak je patrno z tabulky 3.1, konfigurační soubor je rozdělen na části. Jméno každé části (sekce) je uzavřeno v hranatých závorkách. Za jménem následuje seznam atributů, kterým jsou přiřazeny hodnoty pomocí operátoru „=“.

3.5 Používání programu, omezení

Jak jsem se zmínil výše, pro používání programu je třeba nainstalovat si Python a Python shell (viz. 3.2.3 *Python*). Program funguje správně pouze v případě, že je v telefonu nastaven profil *Tichý* – aplikace neumí zatím sama vypnout melodii, která je nastavena v systému a nefungovala by správně.

Nyní už stačí spustit skript `SMSfilter.py` podle obrázku 3.7. V tuto chvíli se načetlo potřebné nastavení pravidel a program čeká na příchozí SMS zprávy.

V případě, že je program spouštěn poprvé, vytvoří se soubory:

- `c:\Data\SMSfilterConfig.txt`,
- `c:\Data\SMSfilter.log`.

Kdykoliv je možné zmáčknout levé softkey tlačítko a objeví se menu, které slouží k přidávání pravidel, jejich zobrazení a smazání nebo k ukončení aplikace.

Pro přidání pravidla postupujeme podle obrázku 3.8.

Otevřeme menu, zvolíme *Add rule*, vybereme si typ pravidla – pro jediný kontakt *Search contacts* nebo pro skupinu kontaktů *Search groups*. V příkladě na obrázku 3.8 jsem zvolil pravidlo pro kontakt. Následně se otevře telefonní seznam, z něhož si vybereme osobu, kterou hodláme filtrovat. Poté následuje výběr akce, v mém případě to byla kontrola obsahu zprávy, jako hledané slovo jsem si zvolil 'Spam'. Nezáleží na tom, zda zadáte slovo s velkými, či malými písmeny, prohledávání je case insensitive – ignoruje velikost písmen. Dále si zvolíme akci, která se provede, pokud je slovo 'Spam' skutečně nalezeno. Já jsem zvolil *Sound ON* – přehraje se melodie, proto je nutné ji zvolit.

Jestliže jste s nastavením spokojeni, stačí už jen odsouhlasit uložení pravidel do konfiguračního souboru a je hotovo.

3.6 Možná rozšíření

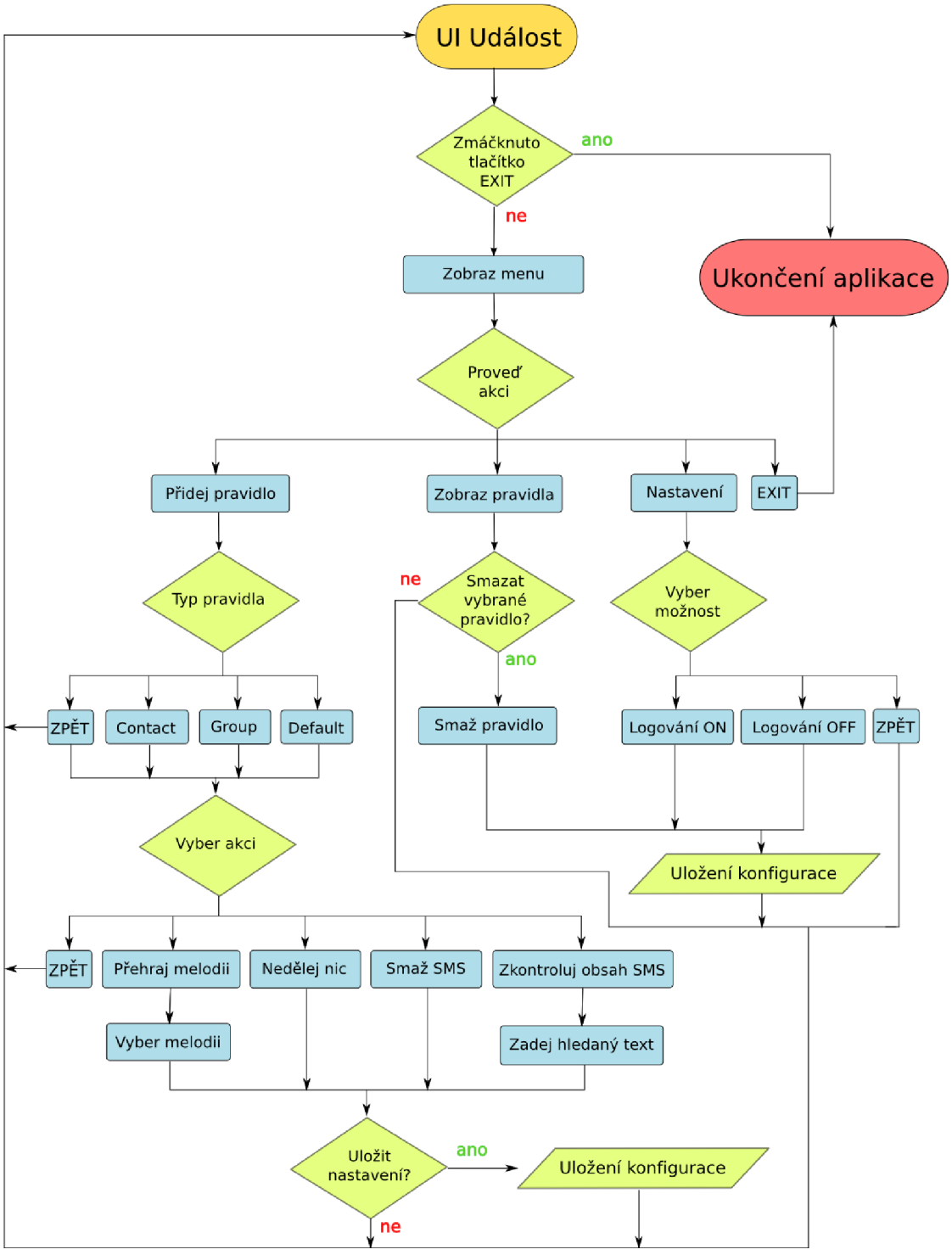
Program není samozřejmě dokonalý a určitě tu jsou věci, které by se daly vylepšit.

Například při zobrazení seznamu pravidel a následném smazání jednoho z nich se seznam neaktualizuje – smazaný kontakt zůstává v seznamu, i když už je fyzicky odstraněn. Za pozornost by stálo i vylepšení způsobu odebírání pravidel a přidání možnosti pro editaci pravidla. V současné verzi je nutno pravidlo nejdříve smazat a pak vytvořit nové.

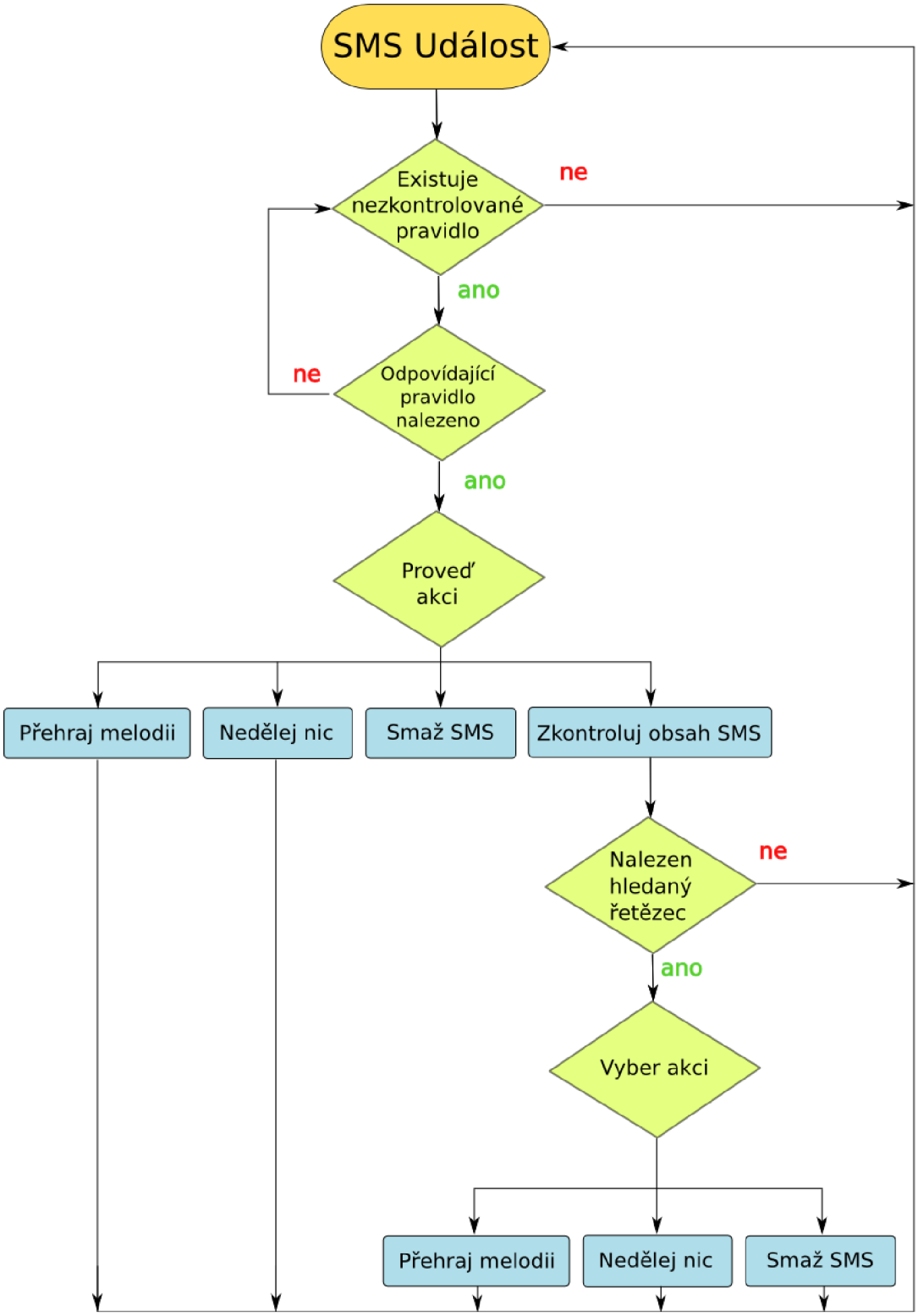
Další z věcí, které by se daly vylepšit by mohla být automatická změna pravidla při změně telefonního čísla (čí jména) v kontaktech – určitá synchronizace s telefonním seznamem.

Také by mohla přijít vhod možnost zadávat výraz pro filtrování obsahu příchozí zprávy ve formě regulárního výrazu.

Samozřejmě nesmím opomenout celkovou optimalizaci programu.



Obrázek 3.5: Schéma části demonstrační aplikace, která zpracovává události uživatelského rozhraní



Obrázek 3.6: Schéma části demonstrační aplikace, která zpracovává příchozí SMS

Obrázek 3.7: Start programu z Python shellu

Obrázek 3.8: Postup při přídávání nového pravidla

Kapitola 4

Závěr

Cílem této bakalářské práce bylo vytvořit aplikaci demonstrující možnosti operačního systému Symbian S60. Tento požadavek byl v rámci bakalářské práce splněn.

Vytvořil jsem aplikaci v programovacím jazyku Python, která dokáže na základě uživatelského nastavení filtrovat příchozí SMS zprávy a reagovat nejen na telefonní číslo odesílatele, ale i na samotný obsah zprávy.

Při hledání na internetu jsem nenalezl žádný podobný open source program.

Původně jsem zamýšlel vytvořit tuto aplikaci v jazyce Java ME, ale kvůli omezením, které jsou s Javou spojeny jsem musel zvolit jazyk jiný, a sice Python. Tato volba se nakonec ukázala jako velmi dobrá, anžto je Python jednoduchý na naučení a je velmi efektivní co se psaní kódu týče.

Program je funkční, samozřejmě existují vždy věci, které by se daly zlepšit. Například by bylo vhodné přidat možnost pro úpravu pravidel, kontrola, zda pravidlo už existuje, či obohatit filtrování obsahu zprávy o možnost filtrovat pomocí regulárního výrazu.

Literatura

[1] COTTET, Francais: *Scheduling in real-time systems*. John Wiley & Sons, 2002, ISBN 0-470-84766-2, anglicky.

[2] DIGIA Inc. (Editor): *Programming for the Series 60 Platform and Symbian OS*. John Wiley & Sons, 2003, ISBN 0-470-84948-7, anglicky.

[3] [online]: Python for S60: Project Filelist. cit [2009-03-12].
URL https://garage.maemo.org/frs/?group_id=854

[4] [online]: Linux a vlákna. cit [2009-03-26].
URL http://www.linux.cz/noviny/1998-0809/clanek11.html

[5] [online]: Wikipedia - Semafor (synchronizace). cit [2009-03-26].
URL http://cs.wikipedia.org/wiki/Semafor_(synchronizace)

[6] [online]: Zdrojový kód třídy FileSel. cit [2009-04-02].
URL http://code.google.com/p/wordmobi/source/browse/trunk/wordmobi/src/filesel.py

[7] [online]: Symbian Foundation. cit [2009-04-04].
URL http://www.symbian.org

[8] [online]: Wikipedia - Symbian Ltd. cit [2009-04-04].
URL http://en.wikipedia.org/wiki/Symbian_Ltd.

[9] [online]: Wikipedia - Symbian OS. cit [2009-04-04].
URL http://en.wikipedia.org/wiki/Symbian_OS

[10] [online]: Introduction to the window server. cit [2009-04-16].
URL http://developer.symbian.com/main/documentation/sdl/symbian94/sdk/doc_source/guide/Graphics-subsystem-guide/WindowServerClientSide/WindowServerGuide1/WindowServerIntroduction.guide.html

[11] [online]: S60 Info. cit [2009-04-29].
URL http://www.s60.com/life/thisiss60/s60indetail/softwareversions

[12] [online]: S60 Info - Flash Lite. cit [2009-04-29].
URL http://www.s60.com/life/thisiss60/S60forbusiness/developers/flashlite/flashlitedetai

[13] [online]: Wikipedia - Java. cit [2009-04-29].
URL http://cs.wikipedia.org/wiki/Java_(programovací_jazyk)

[14] [online]: Open C/C++ - Forum Nokia. cit [2009-05-03].
URL http://www.forum.nokia.com/Resources_and_Information/Explore/Runtime_Platforms/Open_C_and_C++/

[15] [online]: Connected Limited Device Configuration (CLDC). cit [2009-05-06].
URL http://java.sun.com/products/cldc/

[16] [online]: Mobile Information Device Profile (MIDP). cit [2009-05-06].
URL http://java.sun.com/products/midp/

[17] [online]: Python Programming Language – Official Website. cit [2009-05-06].
URL http://www.python.org/

[18] SALES, Jane: *Symbian OS Internals: Real-time Kernel Programming*. John Wiley & Sons, 2005, ISBN 0-470-02524-7, 978-0-470-02524-6, anglicky.

Příloha A

Obsah přiloženého CD

- Adresář *SMSfilter* obsahuje zdrojový kód demonstrační aplikace a software nutný k jejímu běhu na mobilním telefonu se Symbianem.
- Adresář *text* obsahuje tento text zprávy.
- Adresář *latex* obsahuje zdrojové kódy této bakalářské práce v jazyce LaTeXa obrázky.