Sistemas Operacionais – Universidade Federal de Santa Catarina

Sistemas Operativos para Celulares – Symbian OS

Autores

Carlos Jorge Lemos Nunes

Fábio Rafael Magalhães Malheiro

Introdução

Num mundo cada vez mais dependente dos dispositivos móveis, surgiu a necessidade de criar sistemas operativos capazes de lidar, e potenciar, todas as funcionalidades e capacidades oferecidas por estes.

É neste contexto que surge o Symbian OS, um sistema operativo, presente na grande maioria dos principais modelos de *smartphones* actuais, estruturado como os sistemas operativos comuns em desktop, multitarefa preemptivo e com memória protegida.

O desenvolvimento deste sistema operativo assentou em três grandes regras: a integridade e segurança dos dados do utilizador é primordial, o tempo do utilizador não deve ser desperdiçado, e todos os recursos são escassos.Uma vez que o hardware presente num smartphone tem consideráveis limitações de capacidade quando comparado com um PC estas mesmas limitações devem ser tratadas e geridas pelo sistema operativo de forma a disponibilizar todos os recursos de forma optimizada e eficiente ao utilizador.

Com este trabalho e respeitando o âmbito curricular da disciplina de Sistemas Operacionais pretendemos efectuar um estudo das principais características da arquitectura do Symbian OS a nível do controlo de processos, memória, dispositivos de I\O e unidades de armazenamento de dados.

Índice

1.Núcleo

O Symbian OS era baseado no mesmo núcleo do EPOC, o EPOC Kernel 1 (EKA1). O núcleo EKA1, presente no Symbian OS até à versão 7, além de implementar toda a base do sistema operativo, tinha alguns recursos como multi-programação pré-emptiva e protecção de memória.

A partir da versão 8.0 do Symbian OS, passou a ser utilizado um novo núcleo, o EKA2. As principais diferenças e inovações que diferem o EKA2 dos seus antecessores são:

- Sistema de tempo real
- Múltiplos fluxos de execução
- Nanokernel

O suporte do sistema de tempo real no EKA2, implica um tempo limite maior para os processos que exigem a prioridade em tempo real. Ele também faz a gestão dos múltiplos fluxos de execução juntamente com os múltiplos processos. O nanokernel é uma camada mais baixa do núcleo que implementa diversas tarefas num contexto reduzido. No Symbian OS o nanokernel tem como principal função, simplificar a supervisão dos fluxos de execução e exclusões mútuas. Porém, não aceita a alocação de memória dinâmica, sendo apenas uma implementação mais simplória destes conceitos.

O núcleo também é composto por Dynamic Link Libraries (DLL's). São bibliotecas carregadas na memória do sistema, com funções básicas para as aplicações. As DLL's são muito utilizadas no Symbian OS, num total de quase 100 bibliotecas num telefone móvel comum. No Symbiam OS v9.0 toda a interface gráfica do utilizador é baseada nas DLL's. Há dois tipos de DLL, DLL's estáticas e DLL's polimórficas. As DLL's estáticas são colecções de classes e funções básicas do sistema, ou seja, a base de bibliotecas do sistema operativo. As DLL's polimórficas funcionam como plugins para as aplicações.

O núcleo do Symbian OS implementa todas as classes e funções necessárias para o funcionamento sólido do sistema operativo. Ele também provê toda a implementação e gerência de processos e a alocação de memória do sistema.

1.2 Estrutura do Núcleo

A plataforma de um smartphone requer que muitos serviços sejam executados em tempo real, mas ao mesmo tempo tem que lidar com a necessidade de disponibilizar um ambiente de operação e interacção similar a um desktop. Para responder a estes dois requisitos de forma adequada o kernel do Symbian Os tem uma estrutura (Figura 1) mais complexa do que o dos sistemas operativos comuns.

Consoante as necessidades poderão ser efectuados os mais diversificados percursos dentro do kernel. Por exemplo uma aplicação a correr em modo de utilizador, que solicita uma operação no servidor de arquivos. Esta mesma operação irá efectuar um pedido ao kernel que por sua vez irá requisitar I\O em algum dispositivo, que irá fazer uso do nanokernel.

Por sua vez se pensarmos numa chamada telefónica, este tipo de operação irá iniciar as funções presentes no RTOS, que irá interagir directamente com o hardware.

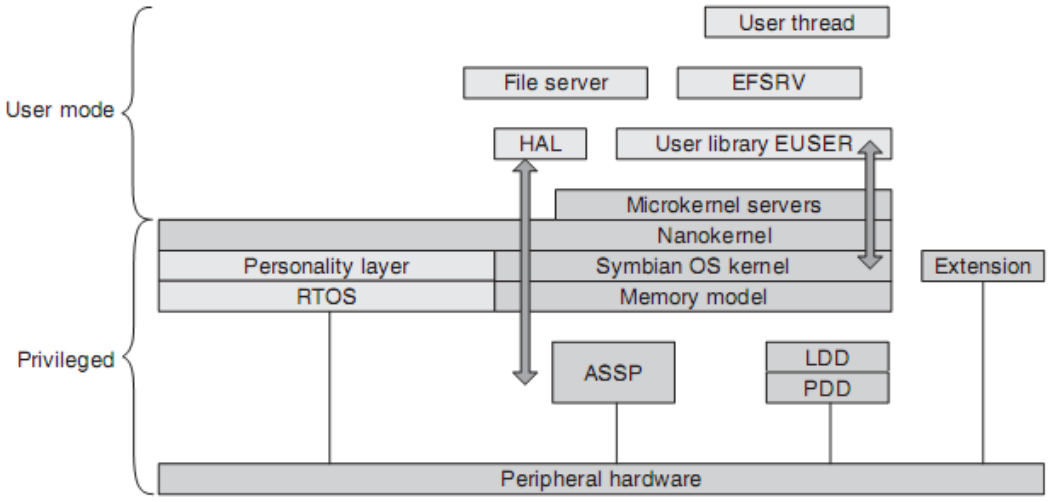

Figura 1 : Estrutura do kernel do Symbian Os

Fonte: Smartphone Operating System Concepts with Symbian OS

- A interface que permite a comunicação entre o código e o hardware está dividida em duas partes:

    1. “Physical device driver” (PDD) que comunica directamente com o hardware.
    2. “Logical Device Driver” (LDD) que estabelece uma interface de comunicação com os níveis superiores de software.

Além disto, o kernel consegue interagir directamente com o hardware através da "application-specific standard product" (ASSP). Por fim, todos os componentes que precisem de interagir com o sistema operativo em tempo real podem interagir directamente com o hardware caso corram num modo especial chamado de "Personality layer"

- ***Memory model***, é um modelo de como a memória esta organizada num dispositivo e como sistema operativo irá funcionar com esta.
- ***Symbian OS Kernel***, depende do *nanokernel* embora esteja separado das partes responsáveis por lidar com tarefas em tempo real. Implementa os diversos modelos de memória exigidos pela plataforma
- ***Nanokernel***, implementa as mais básicas e primitivas partes do Symbian OS e é utilizado maioritariamente pela parte telefónica (responsável pelas envio e recepção de comunicações) do sistema operativo.
- Tanto o "*real-time OS*" como os "*personality layers*" foram desenvolvidos para implementar a funcionalidade telefónica. O "RTOS" implementa as funções de GSM do smartphone através da ligação directa com o hardware. O "personality layer" permite que o fabricante do smartphone use diferentes implementações a nível da função telefónica (como por exemplo função analógica).
- *User mode layers*, inclui os *microkernel servres* tal como as aplicações do utilizador . Tal como referido anteriormente estes interagem com o Symbian Os kernel para efectuarem pedidos ou iniciarem operações em kernel-mode.

2. Processos e Threads

O processo é um programa em execução, seja essa execução uma chamada do próprio sistema ou uma chamada do utilizador. Os sistemas operativos em geral, assim como o Symbian OS, suportam e fazem a gerência da concorrência de processos ou múltiplos processos. O conceito de processo, para Symbian OS, é uma colecção de fluxos de execução que partilham um endereçamento de memória. Essa partilha de memória comum entre processos é limitada no Symbian OS pelo núcleo, por motivos de segurança dos recursos do sistema.

O conceito de fluxo de execução é uma entidade em que o núcleo aloca recursos na UCP. Um processo simples é composto por um conjunto de recursos e um fluxo de execução. Multithreading é a execução de um processo composto por vários fluxos de execução que concorrem entre si baseada em prioridades. O núcleo do Symbian OS disponibiliza o recurso de multithreading, onde vários fluxos de execução podem executar paralelamente sem haver a necessidade do controlo dos fluxos de execução explícito no código em execução.

O nanokernel do Symbian OS suporta os mais simples fluxos de execução do sistema. O sistema também suporta a gerência e sincronização das nanothreads a partir do nanokernel. As nanothreads utilizam uma pilha, única e simples, do administrador do sistema, ou seja, só pode ser executável no sistema no modo administrador.

Os processos e fluxos de execução no Symbian OS, são tratados na forma de multi-programação, a forma já conhecida em sistemas operativos em geral. O sistema possui ainda, internamente no seu núcleo, uma implementação que facilita a gerência dos recursos dos fluxos de execução. Esse recurso basicamente representa uma solução para a limitação dos recursos que esse SO deve tratar.

3. Memoria no Symbian OS

Uma plataforma operada pelo *Symbian OS* oferece os seguintes tipos de memória: *Random Acess Memory* (RAM), *Read Only Memory* (ROM), *Flash Disk* interno e cartões de memória.

A memória RAM é utilizada durante a execução das aplicações, é nela que os programas e o sistema operativo alocam memória durante a execução dos processos e fluxos de execução. Geralmente os dispositivos móveis tem uma quantidade de memória RAM muito pequena.

Parte do sistema operativo *Symbian OS* fica armazenado na memória do tipo ROM, assim como a sua inicialização. A memória ROM é exclusiva do sistema, o utilizador não pode manipular nem escrever dados nessa memória.

O *Flash Disk* é o disco interno para manipulação e escrita de dados pelo utilizador e pelo sistema de arquivos do sistema operativo. O sistema suporta os sistemas de arquivos comuns nos SO's da Microsoft , FAT , e para além deste o sistema de arquivos LFFS (Logging Fast File System).

O núcleo do *Symbian OS* é responsável por alocar a memória física, virtual e gerir os recursos físicos da memória RAM, MMU e caches. A alocação de memória segue uma implementação simples, baseada num heap diferenciado em três partes:

- Uso de memória pré-alocada com tamanho do heap fixo ou segmentação dinâmica do heap.
- Um único fluxo de execução ou múltiplos fluxos de execução com baixo custo do processador.
- Alinhamento selectivo de processos ou fluxos de execução.

Tal como a maioria dos sistemas, o *Symbian OS* divide a memória em páginas lógicas e molduras físicas. O tamanho usual para as molduras é de 4KB. Visto que o tamanho da memória poderá atingir os 4GB (por se tratar de um sistema 32-bit) e tendo em conta o tamanho das molduras de pagina facilmente se percebe que se irá obter uma tabela de página com mais de um milhão de entradas. Devidas às limitações do tamanho de memória, não é viável para o *Symbian OS* dedicar 1MB para a tabela de página, além disso a pesquisa e o acesso a uma tabela de página tão grande provocaria uma enorme sobrecarga ao sistema.

Para resolver isto, o *Symbian OS* adoptou uma tabela de página de dois níveis, conforme demonstrado na figura 2. O primeiro nível, *page directory*, que é indexado pelos primeiros bits do endereço lógico, fornece uma ligação para o segundo nível, que é um conjunto de tabelas de página. Estas tabelas fornecem uma ligação para uma página específica na memória e são indexados pelos 8 bits do meio do endereço lógico. Por fim a página em memória é indexada pelos últimos 12 bits do endereço lógico.

A tradução dos endereços lógicos em endereços físicos é feita por hardware através da MMU e TLB presentes na arquitectura do processador ARM.

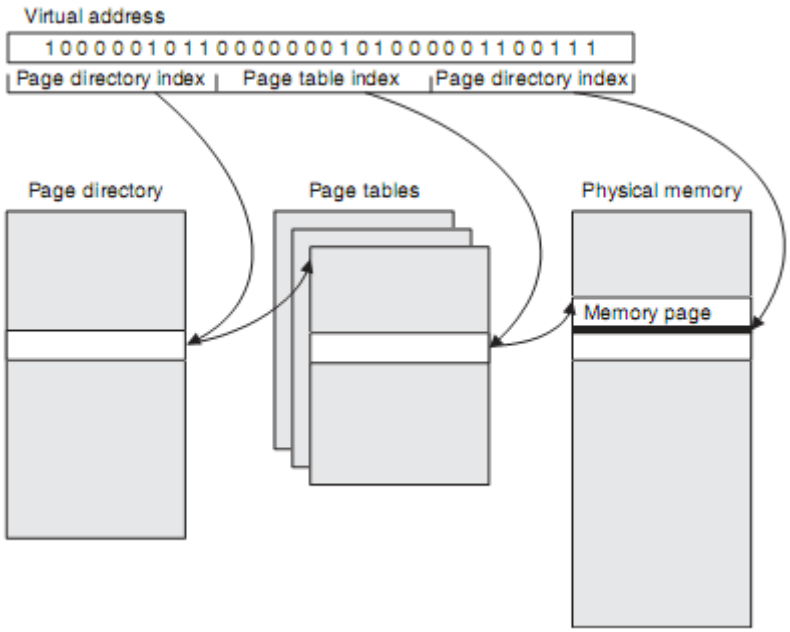

Figura 2 : Tabela de página multi-nível.

Fonte: Smartphone Operating System Concepts with Symbian OS

O que é que acontece quando uma pagina não está presente na memoria?

No *Symbian OS*, isto representa uma condição de erro uma vez que todas as páginas de memória pertencentes a uma aplicação devem ser carregadas no momento em que a aplicação é iniciada. Tendo em atenção que o endereçamento do *Symbian OS* poderá atingir os 4 GB, apesar de ser improvável que um *smartphone* tenha tal quantidade de memória, poderão ocorrer situações em que uma página referenciada não esteja na memória. Tal situação causaria uma "excepção não tratada" de erro, fazendo com que a aplicação do utilizador fosse encerrada.

Molduras de memória são atribuídas a páginas através de uma lista de molduras livres. Se não existirem molduras disponíveis, então ocorre uma situação de erro. Não é

possível substituir molduras de memória usadas com páginas de uma futura aplicação. Mesmo que as molduras pertençam a uma aplicação que já não esteja a executar. Isto acontece porque não existe *swapping* no *Symbian OS* logo não existe um local para onde possam ser copiadas estas páginas "deslocadas".

Há quatro versões diferentes do modelo de implementação de memória no *Symbian OS*. Cada modelo foi projectado para certos tipos de configuração de hardware.

- *Moving Model* – Foi concebido para as arquitecturas mais recentes do ARM. Neste modelo a *page directory* tem 4KB, e cada entrada possui 4 bytes, o que dá um directório de 16 KB. Embora a segmentação não seja explicitamente utilizada existe uma organização para o layout da memória. Existe uma parte para guardar os dados pertencentes ao utilizador, e uma parte para guardar os dados relativamente ao Kernel.

- *Multiple Model* – Modelo concebido para a versão 6 da arquitectura ARM. A MMU nesta versão difere da utilizada nas versões anteriores. Por exemplo, a *page directory*, requer um manuseamento diferente uma vez que pode ser seccionada em duas partes, cada uma referenciando diferentes conjuntos de tabelas de pagina, *user-page tables* e *kernel-page tables*.

- *Direct Model* – Este modelo assume que não existe uma MMU. É um modelo raramente utilizado uma vez que a falta de MMU poderia causar graves problemas de performance. Este modelo é útil para ambientes de desenvolvimento em que por alguma razão seja útil ter a MMU desactivada

- *Emulator Model* – Foi desenvolvido para suportar o emulador do *Symbian OS* no *Microsoft Windows*. Tal como seria de esperar este modelo apresenta algumas restrições. O emulador corre como um único processo no Microsoft Windows. O espaço de endereçamento está limitado a 2GB (ao invés dos 4GB). Toda a memoria disponibilizada para o emulador é acessível por qualquer processo do *Symbian OS*. Não existe protecção de memória. As bibliotecas do *Symbian OS* são disponibilizadas como DLL's do *Microsoft Windows* , cabendo a este manusear a sua alocação e respectiva gestão de memória.

4. Gerência de Processador

Os celulares têm sua arquitectura de hardware baseada em microprocessadores assíncronos ARM (Advanced RISC Machine) de 32bits. As características principais destes microprocessadores são o tamanho reduzido e o baixo consumo de energia, o que corresponde com os requisitos do Symbian OS. O desempenho alto exigido pelo Symbian OS v9.3 é relativo, entre 100 e 200MHz, quando comparado com os CPUs dos desktops que chegam a 3GHz. As versões mais recentes, baseadas no EKA2, do Symbian OS são projectadas para o microprocessador ARM modelo v5T. Um processo em Symbian OS é um executável que tem o seu próprio espaço na memória, uma pilha e um cabeçalho. O Symbian OS também suporta múltiplos processos e múltiplos fluxos de execução. Todo o processo tem um nome associado ao mesmo, e por padrão é utilizado o nome do arquivo .exe em questão.

A partir da oitava versão do sistema que utiliza o EKA2, o Symbian OS inova o tratamento de multi-programação. Para tratar os processos e fluxos de execução utiliza escalonadores tradicionais, que são controlados pelo nanokernel. Alguns processos são considerados de alta prioridade, como o controlo das chamadas telefónicas, aplicações de vídeo e música, animações e outras aplicações que usam protocolos de comunicação. A gerência desses processos é implementada com um FIFO, onde os primeiros processos têm a prioridade mais alta da fila. O escalonamento dos fluxos de execução, seguem essa linha híbrida, onde para cada tipo de processo ou fluxo de execução o sistema implementa um método para gerenciar os seus recursos. O tratamento dos múltiplos fluxos de execução, são tratados no Symbian OS com um escalonador Round-Robin com prioridades (64níveis). O nanokernel implementa, internamente, algoritmos escalonadores Round-Robin com prioridades, limite de tempo de execução e semáforos.

A prioridade entre os processos é tratada pelo núcleo, seguindo a prioridade dada a cada processo, seja pelo próprio sistema, seja pelo utilizador, ou seja, pelo programador. O processo chega ao estado final quando termina a sua execução normal ou num processo de kill, quando o processo é encerrado por um outro motivo indicado pelo sistema operativo através de um número inteiro. Neste caso, o processo pode ser interrompido e finalizado de duas formas: kill itself, quando ele encerra a sua execução ou quando ele é encerrado por meio de outro processo. Os processos nesse sistema operativo são tratados internamente no núcleo, ou num nível mais baixo, no nanokernel. Este último, implementa alguns escalonadores tradicionais como o FIFO, Round-Robin e semáforos para controlar a fila de processos e fluxos de execução.

5. Entrada e Saída

O hardware do sistema como um todo, possui diversos periféricos conectados através de uma interface. Essa interface constitui num elemento fundamental para a comunicação do sistema operativo com esses periféricos, e para isto, usam-se os controladores. O controlador funciona na forma de controlar cada periférico, no intuito de realizar e fazer funcionar as tarefas deste periférico. Cada periférico deve ter um controlador projectado para fazê-lo funcionar. O controlador é responsável por fazer interagir o hardware e o software, no caso o sistema operativo, através das interfaces dos dispositivos.

Para fazer a gerência dos processos de entrada e saída, existem diversas técnicas como E/S programada, interrupções, e acesso directo à memória (DMA). O Symbian OS utiliza duas das tradicionais técnicas, interrupções e DMA. O mecanismo de interrupções utilizado no Symbian OS, determina as interrupções no kernel a partir dos sinais emitidos em dois registadores do tipo PIC. Essas interrupções são divididas de duas formas: Interrupt Request (IRQ), ou interrupção normal, e Fast Interrupt Request (FIQ), sendo esta última com maior prioridade sobre as IRQ's. Essa organização das interrupções pode ser vista na Figura 3.1

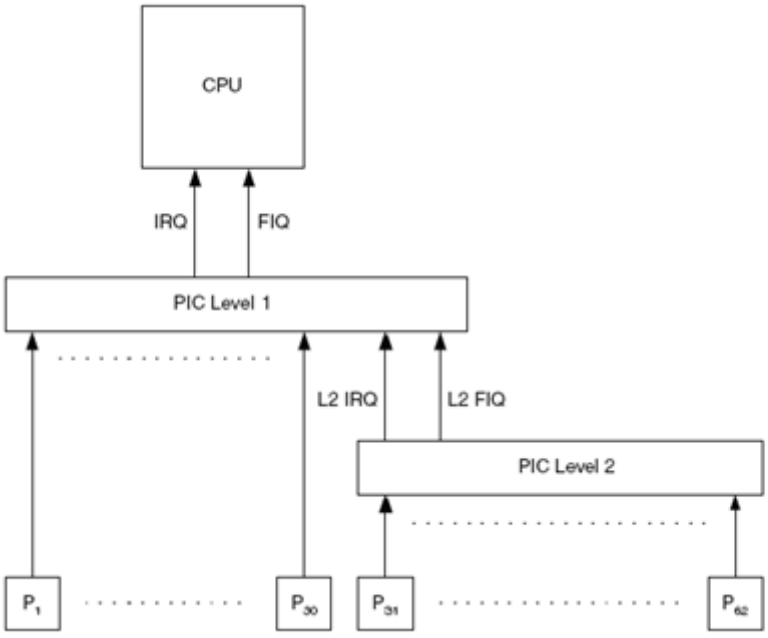

Figura 3.1 - Funcionamento das Interrupções no Symbian OS.

De acordo com a Figura 3.1, o mecanismo de interrupções do sistema tem dois níveis, separados pelos registadores PIC. O nível 1 (PIC Level 1) tem acesso directo ao processador (CPU). Já no nível 2 (PIC Level 2) existe uma ponte entre o nível 1 e o processador, para acontecer a interrupção do processo de E/S.

Como o mecanismo de interrupções não é sempre eficiente, os sistemas utilizam a técnica de DMA, ou acesso directo à memória. Não diferente dos conceitos básicos de SO's o Symbian OS utiliza também desse mecanismo para agilizar os processos de E/S. A técnica de DMA permite aos periféricos e/ou dispositivos alocar memória directamente sem que exista um processo no CPU para isto. No Symbian OS é comum o uso de DMA em multicanais, onde é permitida, mais de uma transferência de dados de E/S na memória, divididos em canais. Um canal do DMA inicializa o processo de transferência de dados pelo registador do periférico e endereça memória directamente. Os canais são organizados em uma FIFO e controlados pela interface de cada periférico.

Na interacção hardware e software, o sistema operativo necessita de controladores dos dispositivos, onde na visão do sistema operativo aplica-se o conceito de drivers de dispositivos ou device drivers. Drivers de dispositivos são um meio das aplicações terem acesso aos recursos dos periféricos através do sistema operativo, sem que seja necessário ao programador implementar códigos de baixo nível para fazer uso desses periféricos.

Os drivers de dispositivos são carregados no núcleo do Symbian OS através das DLL's de um modo dinâmico. A arquitectura desses drivers é divida em duas partes: Logical Device Driver (LDD) e Physical Device Driver (PDD). As aplicações podem utilizar os drivers mais comuns directamente da biblioteca LDD ou caso seja necessário tem o suporte da PDD. A PDD é uma opção na arquitectura dos drivers, e funciona na comunicação somente do hardware com a LDD, ou seja, inatingível para aplicação directamente.

No Symbian OS há também extensões para o controlo de outros dispositivos, que já estão compilados nativamente no núcleo do sistema. O subsistema de camadas do Symbian OS e do EKA2 está organizado em diversos blocos modulares, não monolíticos. Algumas camadas associadas ao EKA2, podem ser vistas na Figura 3.2.. O sistema também é projectado para um único utilizador, diferente dos SO's para PC, como Microsoft Windows, GNU/Linux e Apple Mac OS X.

Figura 3.2 – Controlo de dispositivos pelo EKA2

O Symbian OS também é baseado numa arquitectura GUI (interface gráfica do utilizador), com diversos recursos muito flexíveis.

Os periféricos disponíveis no hardware dos dispositivos têm que ser geridos, e têm que estar disponíveis para as aplicações e para o utilizador. O controlo de E/S do sistema operativo entra na ponte entre esses dispositivos e o suporte e funcionamento deles para o sistema como um todo.

BIBLIOGRAFIA

http://en.wikipedia.org/wiki/Symbian_OS

http://en.wikipedia.org/wiki/Active_objects

Lucas Simões de Carvalho, Marcel Cunha Batista, Vinicius Ulbrich ,Análise de ferramentas para o desenvolvimento de aplicações para Sistema Operacional Symbian.

Michael J. Jipping , Smartphone Operating System Concepts with Symbian