iOS

Kesley Kenny Vasques Guimarães*, Pedro Henrique de Brito Agnes†
Universidade de Brasília, Departmento de Ciência da Computação
Brasília, Brasil
Email: {180021231, 180026305}@aluno.unb.br

***Resumo*—Nesse artigo serão apresentados aspectos técnicos do sistema operacional dos dispositivos móveis da Apple, o iOS. Serão observadas diversas características de acordo com os conceitos estudados em sistemas operacionais, como a forma em que a gerência de processos funciona, classificando-a como monolítica, micro-núcleo, entre outros. Também veremos sobre a gerência de memória dos dispositivos, abordado tópicos de memória virtual como a paginação, sobre a arquitetura, como o sistema divide os seus módulos de forma a atribuir responsabilidades diferentes a cada um, entre outros exemplos. Por fim, serão avaliados os aspectos para apontar vantagens e desvantagens das implementações utilizadas no iOS.**

***Index Terms*—iOS, Apple, Sistema operacional, Arquitetura, Gerência, Kernel, Threads.**

I. Introdução

O iOS (iPhone *Operating System*) é o sistema operacional utilizado nos dispositivos móveis da Apple, se responsabilizando pelo gerenciamento nos dispositivos como iPhone, iPod e iPad. O desenvolvimento do iOS foi feito, a primeira vista, exclusivamente para o celular da Apple, o iPhone, tendo a data de lançamento juntamente com a primeira versão do iPhone em 2007. Desde então, o sistema foi sendo aprimorado e distribuído exclusivamente aos outros dispositivos móveis proprietários da empresa, nunca sendo distribuído para utilização em outras marcas.

Atualmente, o iOS é um dos sistemas operacionais de dispositivos móveis mais utilizados, apesar de sua exclusividade à marca, pois os iPhones estão sempre entre os celulares mais vendidos no mundo. O principal concorrente do iOS é o Android, que atualmente se encontra no topo do mercado *mobile* por ser open-source e gratuito, sendo usado na maioria dos aparelhos, independente de marca.

II. Especificações

Todo sistema operacional moderno deve ter as funções principais de gerenciar os recursos disponíveis na máquina e abstrair a complexidade de funcionamento da mesma para permitir que qualquer usuário consiga utilizá-la sem conhecimentos muito avançados. Para isso, eles devem fazer a gerência de processos, arquivos, memória, entrada e saída, além de outras especificações como o kernel, funcionamento das interrupções e threads. De forma resumida, um sistema operacional deve atuar como um intermediário entre o *software* e o *hardware* e é exatamente essa função que o iOS exerce nos dispositivos móveis da Apple.

A. Arquitetura

A arquitetura do iOS é composta por 4 camadas que oferecem *frameworks* e serviços a camadas superiores e para serem utilizados pelas aplicações para o acesso de recursos do iPhone, de forma que os processos não os acessem diretamente. A camada mais superior é a *Cocoa Touch*, que oferece serviços mais sofisticados como multitarefa e serviços de notificação. É a camada de mais alto nível, tendo como principal objetivo, disponibilizar recursos para os programadores desenvolverem para a arquitetura.

Na segunda camada, temos a camada *Media*, que basicamente contém as tecnologias de áudio, gráfico e vídeo, que são disponibilizadas ao desenvolvedor para facilitar a criação de aplicativos *multimedia*. Já na terceira camada, temos os *Core Services*, que agrupa vários serviços fundamentais do sistema que também são usados pelos programadores no desenvolvimento para a plataforma. Essa camada possui um módulo que utiliza conceitos de programação assíncrona e núcleo otimizado para oferecer alternativas para o *threading*. Ainda nessa camada, também existem outras funções de mais baixo nível, como o suporte para leitura e escrita de arquivos e até um banco de dados local para ser utilizado pelas aplicações.

Por fim, temos a quarta e última camada, chamada de *Core OS*, que é o nível do sistema e onde encontramos as interfaces de mais baixo nível da arquitetura, assim como o kernel e *drivers*. Dentre as interfaces da camada, estão as de *threading*, que é o suporte a *threads*, *networking* que permite o acesso à rede, também tem o acesso ao sistema de arquivos, gerenciamento de entrada e saída e gerenciamento de memória.

B. Gerência de Memória

Para o gerenciamento de memória no iOS, primeiramente os blocos de memória são divididos em três tipos: *Clean*, *Dirty* e *Free*, que indicam respectivamente os blocos que foram usados por processos anteriormente, mas foram liberados e estão disponíveis para uso, os blocos ocupados no momento e os blocos que nunca foram ocupados. Assim o gerenciador consegue avaliar o espaço disponível na hora de alocar, assim como evitar a fragmentação, onde o sistema tende a alocar no espaço descrito por *Clean* quando disponível.

O iOS suporta o conceito de memória virtual, onde os processos podem ser movidos entre a memória principal e secundária por meio do *swapping*. O iOS suporta a paginação e o tamanho das páginas é de 16 kilobytes em versões mais recentes e é utilizada uma tabela de páginas para o gerenciamento delas. Porém, apesar de suportar a prática, ela

não é muito utilizada por escolha de *design* de forma a buscar a melhor performance. Apenas as páginas definidas como apenas leitura podem realizar o *swapping* e em relação aos processos de usuário, ao invés de depender da paginação, o sistema mata os que não estão sendo utilizados a mais tempo para liberar espaço para novos, o que força os processos a reiniciarem por completo caso entrem na fila de execução novamente.

O iOS conta também com uma estrutura de alto nível que funciona como *garbage collector* já nativa do sistema, chamado de ARC (*Automatic Reference Counting*). Esta estrutura vai verificar se os itens na memória principal possuem referências apontando para si, e no caso de não existirem, automaticamente elimina esses itens da memória para liberar espaço, o que traz consequências positivas no gerenciamento de memória.

C. Gerência de Processos

O iOS permite que múltiplos processos operem simultaneamente na memória principal, realizando também o compartilhamento de tempo (time-sharing). Esse conceito é denominado multiprogramação. O escalonador, subsistema do sistema operacional, é o principal responsável por decidir qual o processo tomará a posse da CPU e qual processo deverá aguardar. O algoritmo de escalonamento utilizado nos sistemas iOS é o de múltiplas filas com realimentação, onde cada grupo de filas é agrupado de acordo com suas características e prioridades.

Cada aplicativo é um processo por si só e atua sobre um kernel monolítico. O aplicativo, após iniciado, pode fazer solicitações por mais recursos, gerando novos processos que podem ou não estarem em execução. Cada processo possui um estado no seu contexto de aplicação. Existe o total de 5 estados, que podem ser definidos em: não executando, inativo, ativo, background e suspenso.

D. Gerência de Arquivos

O APFS (Apple File System) é o sistema responsável por gerenciar os arquivos utilizados no IOS. Foi criado em 2017 com objetivo de substituir o HFS+ (antigo gerenciador de arquivos). O AFPS trouxe com ele proporções maiores de desempenho em memórias flash. Além disso, recursos como clonagem, cópias de arquivos sem que haja uma maior ocupação do espaçamento original e snapshot, serviço que fornece arquivos de leitura com rapidez e segurança durante o processo de recuperação dos arquivos também foram implementados.

O sistema também conta com compartilhamento de espaços alocados, eliminação de necessidade de partições fixas, utilização de contêineres com alocação dinâmica, dimensionamento rápido de diretórios, criptografia robusta e maior capacidade de armazenamento de arquivos.

E. Gerência de E/S

O componente responsável pela alocação dos drivers dos dispositivos de E/S em sistemas XNU é o framework I/O Kit, escrito em C++. Ele provê abstrações do hardware do sistema com classes base pré-definidas, fazendo com que a implementação da maioria dos novos dispositivos seja plug & play. Além disso, cada dispositivo pertence a um controller correlacionado a ele, por exemplo, um dispositivo fone herda atributos da classe fone, sendo necessário somente adicionar novos drivers de instalação (caso exista). Essa abordagem é bastante otimizada, pois há reaproveitamento de código entre os drivers dos dispositivos.

O I/O Kit conta com duas bases de dados, a base de catálogo dos registros de todas as classes disponíveis para uso e outra base que marca a movimentação dos objetos instanciados por essas classes. Cada objeto normalmente representa um dispositivo sendo utilizado.

F. Interrupções

O iOS suporta interrupções, de forma que as threads de menor prioridade podem ser preemptadas por processos do sistema. A rotina de tratamento de interrupções é chamada quando *Mach Exceptions* ocorrem, que vai tratar das exceções quando eventos de caráter excepcional interrompem threads em execução e, dependendo do tipo de exceção, a tarefa pode voltar a executar após o tratamento. O próprio sistema disponibiliza ferramentas de alto nível para permitir ao programador alterar a prioridade das threads criadas pelo processo, o que pode impedir uma preempção indesejada no caso de aumentar essa prioridade com sucesso.

Dentre os exemplos de interrupção, estão as *system calls* e outros processos do sistema, porém alguns processos de usuário podem também causar interrupções, como quando precisam utilizar recursos de bem baixo nível. Uma coisa que é importante notar, é que o sistema de sinais utilizado pelo iOS funciona em cima do sistema de exceções utilizado para interrupções.

G. Kernel e Threads

O Kernel do iOS utiliza uma arquitetura híbrida de monolítico para micro-núcleo vinda da época em que a Apple comprou a empresa NeXT fundada por Steve Jobs, e assim foi criado o XNU. A arquitetura tem a vantagem de ser quase tão rápida quanto a monolítica, porém mais segura e confiável usando características de micro-núcleo, de forma a não derrubar o sistema inteiro se um dispositivo com driver defeituoso for conectado ao sistema por exemplo.

O iOS é um sistema multi-thread, onde os processos podem possuir múltiplas threads. Todo processo iniciado no iOS começa com apenas uma única thread, que é a principal e pode ou não criar novas threads que, apesar de independente da principal, serão vistas como uma só pelo sistema e compartilham o mesmo espaço alocado de memória. Na camada de *Core Services* (Seção II-A), existe um módulo chamado *Grand Central Dispatch*, que abstrai os detalhes mais internos da utilização de threads para o programador e permite a implementação de concorrência, assim como permitindo funções para sincronia das threads, atuando como uma estrutura conhecida como monitor.

III. Conclusões

A tendência para os sistemas operacionais da atualidade é rodar o mínimo possível em modo kernel, e mais em modo usuário, que tem sido uma transição um tanto demorada, que porém está acontecendo. O iOS possui ainda características vindas de versões anteriores do MAC OS, de sistemas monolíticos, que utilizam o modo kernel em tudo, porém é híbrido com sistemas micro-núcleo, onde é percebida uma maior segurança. Outro aspecto é sobre a memória, que apesar de implementar a paginação, não a utiliza muito, assim como em outros dispositivos móveis, que os processos em segundo plano tentem a ser totalmente eliminados pelo sistema, ao invés de realizar um *swapping*.

De modo geral, os dispositivos que utilizam como base os sistemas iOS estão desempenhando avanços significativos na comunidade de desenvolvimento de softwares (aportuguesamento morfológico) básicos. Com uma arquitetura bastante robusta sendo utilizada e algoritmos de gerenciamento de hardware otimizados, há uma maior capacidade de aproveitamento dos recursos que um dispositivo móvel fornece ao seu usuário.

Referências

[1] Tanenbaum, A. S., Bos, W. *Sistemas Operacionais Modernos*. Person, São Paulo, 4° ed., 2016

[2] Adriano Mendonça Rocha, Roberto Mendes Finzi Neto, *Introdução à Arquitetura Apple iOS*, 2011. https://www.enacomp.com.br/biblioteca-de-publicacoes/get-file.php?id=5929

[3] Apple, I. (2008). *iPhone OS Programming Guide*. Apple Inc.

[4] Apple, I. (2010). *iOS Technology Overview*. Apple Inc.

[5] Apple *Memory Usage Performance Guidelines*, 2013. https://developer.apple.com/library/archive/documentation/Performance/Conceptual/ManagingMemory/Articles/AboutMemory.html

[6] Apple *Kernel Programming Guide*, 2013. https://developer.apple.com/library/archive/documentation/Darwin/Conceptual/KernelProgramming/About/About.html#//apple_ref/doc/uid/TP30000905

[7] Niel Lebeck, Arvind Krishnamurthy, Henry M. Levy, Irene Zhang. *End the Senseless Killing: Improving Memory Management for Mobile Operating Systems*, 2019. https://dada.cs.washington.edu/research/tr/2019/04/UW-CSE-19-04-01.pdf

[8] https://www.quora.com/Are-iOS-operating-systems-based-on-Linux

[9] https://pt.wikipedia.org/wiki/Steve_Jobs

[10] https://ronaldolima.eti.br/threads-e-ios-3e1cbe9fe1a

[11] Apple *Using the Mach Thread API to Influence Scheduling*, https://developer.apple.com/library/archive/documentation/Darwin/Conceptual/KernelProgramming/scheduler/scheduler.html#//apple_ref/doc/uid/TP30000905-CH211-BABHGEFA

[12] Ole Henry Halvorsen, *OS X and iOS Kernel Programming* http://3.droppdf.com/files/mJdlB/os-x-and-ios-kernel-programming.pdf

[13] Diego Melo, *O que é APFS?*. https://tecnoblog.net/358465/o-que-e-apfs-apple-file-system/

[14] Saumil Shulka, Richa Mishra, Rishabh Bidya, *Process Management - iOS*. https://prezi.com/cna3bm7eqqdc/ios-process-management-rishabh-saumil-richa/

[15] https://slideplayer.com.br/slide/10639632/