INSTITUTO FEDERAL DE SÃO PAULO
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

ANDREY GUSTAVO SEGANTIN DOS SANTOS

ELTON LIMA ARAUJO

JOÃO GUSTAVO DOS SANTOS

SISTEMA OPERACIONAL MACOS

SÃO PAULO

2025

1. INTRODUÇÃO

O sistema operacional é uma parte crucial em um dispositivo computacional. Ele é responsável por conectar o software (parte lógica, sendo programas que fazem o computador funcionar) e o hardware (componentes físicos), além de gerar ao usuário uma interface intuitiva que facilite a utilização do sistema.

Atualmente, existem vários sistemas operacionais de diferentes tipos, sendo os mais conhecidos o Windows, macOS e as diferentes distribuições de Linux, como o Ubuntu. Entre essas, o sistema dos dispositivos Mac, da empresa Apple, se mostra uma opção segura, estável e com uma interface fácil de usar, gerando constantemente novas atualizações que melhoram o sistema a cada versão.

Dessa forma, esse trabalho tem a finalidade de explicar como o macOS funciona, analisando alguns elementos como o Kernel, sistema de arquivo, gerenciamento de memória e gerenciamento de processos.

2. METODOLOGIA

A metodologia utilizada para a realização da pesquisa sobre o tema foi qualitativa, utilizando sites considerados confiáveis, como o site oficial da Apple. Dessa forma, foi possível realizar um trabalho que investigasse o funcionamento do Kernel, sistemas de arquivos, e outras partes do sistema operacional utilizado nos dispositivos Mac.

Para a realização da pesquisa, foram utilizadas diferentes versões do sistema, isso porque o macOS é, em sua grande parte, código fechado, o que impossibilita uma análise profunda do funcionamento de “releases” específicas. No tópico do Kernel, foi utilizada a documentação oficial do site da Apple, onde a versão utilizada para a elaboração do conteúdo não é especificada. Porém, é possível acreditar que a documentação se referia à uma versão anterior ao rebranding do ”OS X” para o “macOS”, no ano de 2016. O texto parece se referir à uma versão entre o OS X 10.0 (Cheetah, de 2001), e OS X 10.10 (Yosemite, de 2014), sendo em sua fase inicial ou intermediária.

Assim, essa pesquisa tem o objetivo de esclarecer o funcionamento do macOS, não por completo, uma vez que não se trata de um sistema totalmente “open source”, mas de uma forma que gere uma análise satisfatória e coerente.

3. HISTÓRIA DO MACOS

O sistema operacional da Apple teve seu início em 1976, com o Apple I, que não tinha em si um sistema operacional tradicional. O Apple II, lançado em 1977 trouxe o Apple DOS, o que permitiu a execução de programas e gerenciamento de arquivos.

No ano de 1984, o Macintosh foi lançado, revolucionando a computação com o Mac OS, um dos primeiros sistemas operacionais com interface gráfica. A Apple, mesmo em meio a inovações, enfrentava dificuldades no mercado. Em 1985, após ser demitido da Apple, Steve Jobs, fundou a NeXT e desenvolveu o NeXTSTEP, um sistema avançado baseado em Mach e BSD.

3.1. OS X

Após isso, em 1997, a Apple comprou a NeXT e o Steve Jobs retornou à empresa. O sistema NeXTSTEP serviu como base para o Mac OS X, lançado em 2001. Essas são as versões do OS X:

1. OS X 10.0 "Cheetah" (2001)
2. OS X 10.1 "Puma" (2001)
3. OS X 10.2 "Jaguar" (2002)
4. OS X 10.3 "Panther" (2003)
5. OS X 10.4 "Tiger" (2005)
6. OS X 10.5 "Leopard" (2007)
7. OS X 10.6 "Snow Leopard" (2009)
8. OS X 10.7 "Lion" (2011)
9. OS X 10.8 "Mountain Lion" (2012)
10. OS X 10.9 "Mavericks" (2013)
11. OS X 10.10 "Yosemite" (2014)
12. OS X 10.11 "El Capitan" (2015)

3.2. macOS

No ano de 2016, ocorreu o lançamento do macOS 10.12 "Sierra", mudando sua nomenclatura para se alinhar com outros sistemas da Apple, como o IOS. Estas são as versões do macOS:

1. macOS 10.12 "Sierra" (2016)
2. macOS 10.13 "High Sierra" (2017)
3. macOS 10.14 "Mojave" (2018)
4. macOS 10.15 "Catalina" (2019)
5. macOS 11 "Big Sur" (2020)
6. macOS 12 "Monterey" (2021)
7. macOS 13 "Ventura" (2022)
8. macOS 14 "Sonoma" (2023)

4. ESTRUTURA DO KERNEL

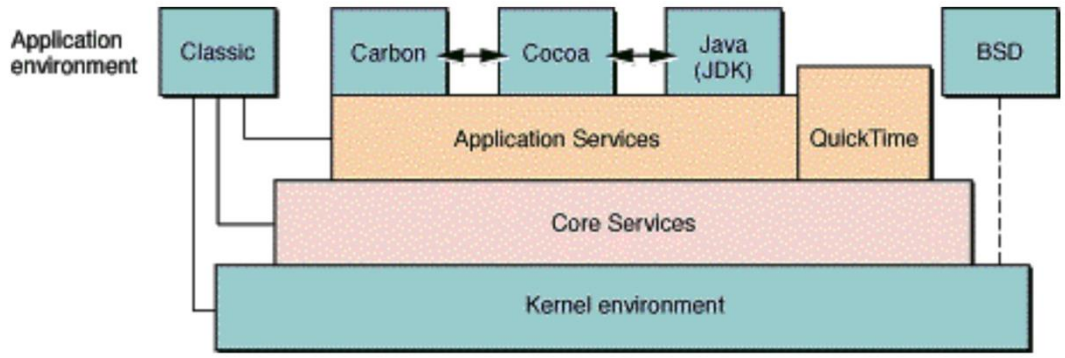
O kernel é responsável por fornecer funções essenciais, como desempenho otimizado, suporte a sistemas de arquivo, APIs orientada a objetos, além de proteção de memória e preempção, que contribui para a segurança e robustez do sistema.

No sistema operacional Mac OS 9, de 1999, a estabilidade e capacidade do sistema eram afetadas quando um aplicativo não segue as regras de cooperação necessárias, haja vista que funcionavam em um ambiente de multitarefa cooperativa, onde a memória do sistema e o tempo de processamentos eram compartilhados.

No entanto, o OS X utilizou uma multitarefa preemptiva, onde a alocação do processador e da memória eram controlados pelo núcleo do SO de forma eficiente, garantindo um melhor desempenho em relação ao modelo anterior.

Ademais, no OS X, cada processo possui seu próprio espaço de endereço de memória e é protegido pelo kernel. Logo, isso impede que um aplicativo acesse ou modifique acidentalmente a memória de outro. O sistema também oferece diferentes formas de comunicação entre os processos, como bibliotecas compartilhadas, frameworks, memória compartilhada POSIX e o sistema de mensagens Mach.

Figura 1 - Arquitetura do Sistema Operacional macOS



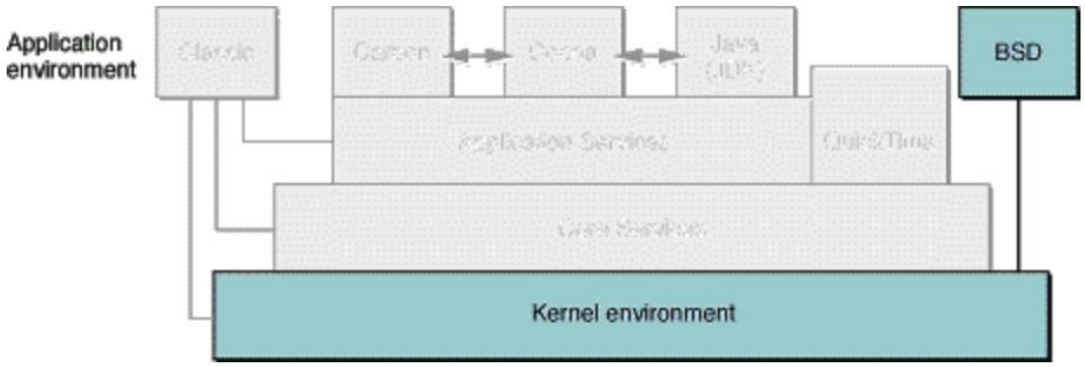
Fonte: https://developer.apple.com/library/archive/documentation/Darwin/Conceptual/KernelProgramming/art/osxlayers.gif

4.1 Darwin

O kernel do OS X faz parte do Darwin, que é um sistema operacional completo, sendo ele opensource e baseado em tecnologias como BSD e Mach 3.0. Porém, nele, não são incluídas camadas gráficas como Quartz (responsável pelo sistema de renderização gráfica do macOS), QuickTime (estrutura de mídia usada para reproduzir, editar e codificar áudio e vídeo), Cocoa (framework de desenvolvimento de aplicativos nativos para macOS), Carbon (permite que aplicativos anteriores ao OS X fossem portados para o novo sistema operacional) e o OpenGL (API de renderização gráfica 2D e 3D).

O OS X e o Darwin compartilham o mesmo núcleo, sendo o Darwin baseado em tecnologias como o FreeBSD. O OS X inclui algumas camadas como o QuickTime, Application Services, Core Services e ambientes como o Carbon, Cocoa e Classic.

Figura 2 – Estrutura do Darwin

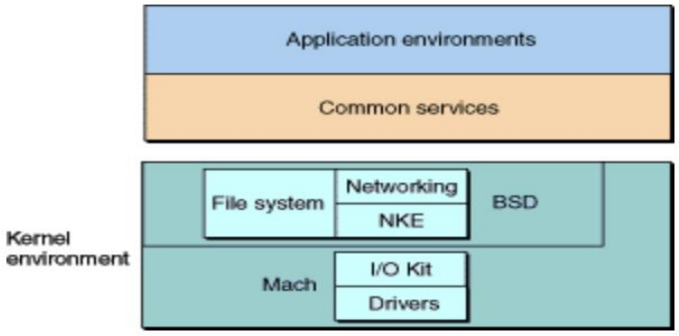


Fonte: https://developer.apple.com/library/archive/documentation/Darwin/Conceptual/KernelProgramming/art/darwinlayers.gif

4.2 Componentes da Arquitetura do Kernel

A arquitetura do kernel do OS X é constituída por três partes principais, sendo elas o Mach, BSD e o I/O Kit.

Figura 3 - Arquitetura do Kernel



Fonte: https://developer.apple.com/library/archive/documentation/Darwin/Conceptual/KernelProgramming/art/osxarchitecture.gif

4.3. Mach

O Mach é responsável por gerenciar recursos críticos do sistema, como uso de CPU, agendamento de processos e memória, fornecendo uma infraestrutura centrada

em mensagens para comunicação entre processos. Ele oferece uma série de funcionalidades incluindo:

- Arquitetura altamente modular: responsável por facilitar a extensão do sistema.
- Comunicação entre processos não tipada (IPC): Permite que processos se comuniquem de uma maneira segura e eficiente.
- Chamadas de procedimento remoto (RPC): Facilita a execução de funções em processos remotos.
- Suporte ao planejador para multiprocessamento simétrico (SMP): Possibilita, ao sistema, a utilização de múltiplos processadores.
- Suporte para serviços em tempo real: Oferece garantias de tempo de resposta para aplicativos críticos.
- Suporte de memória virtual: Responsável por gerir a e proteção q alocação de memória para cada processo.
- Suporte para pagers: Permite um gerenciamento dinâmico da memória virtual.

4.4. BSD

Acima da camada Mach, o BSD fornece serviços que constituem a “personalidade do sistema operacional”. Baseado no núcleo do FreeBSD (uma versão do 4.4BSD), ele possibilita ao SO a utilização de recursos de rede, segurança e compatibilidade. Ele oferece várias funcionalidades como:

- Sistemas de arquivos;
- Redes;
- Modelo de segurança UNIX;
- Suporte a syscalls;
- Modelo de processo BSD;
- APIs do kernel do FreeBSD;
- APIs POSIX;
- Suporte a pthreads (threads POSIX).

4.5. I/O Kit

O I/O Kit, contido dentro do Mach, disponibiliza uma estutura que simplifica o desenvolvimento dos drivers, fornecendo suporte a diferentes tipos de dispositivos. Sua arquitetura de I/O (entrada e saída) é orientada a objetos, desenvolvida em um subconjunto restrito a C++. Além disso, sua estrutura é altamente modular, o que permite uma flexibilidade maior no desenvolvimento dos drivers. Ele oferece:

- Plug and play verdadeiro;
- Gerenciamento dinâmico de dispositivos;

- Carregamento dinâmico (“sob demanda”) de drivers;
- Gerenciamento de energia para sistemas de desktop e portáteis;
- Capacidades multiprocessador.

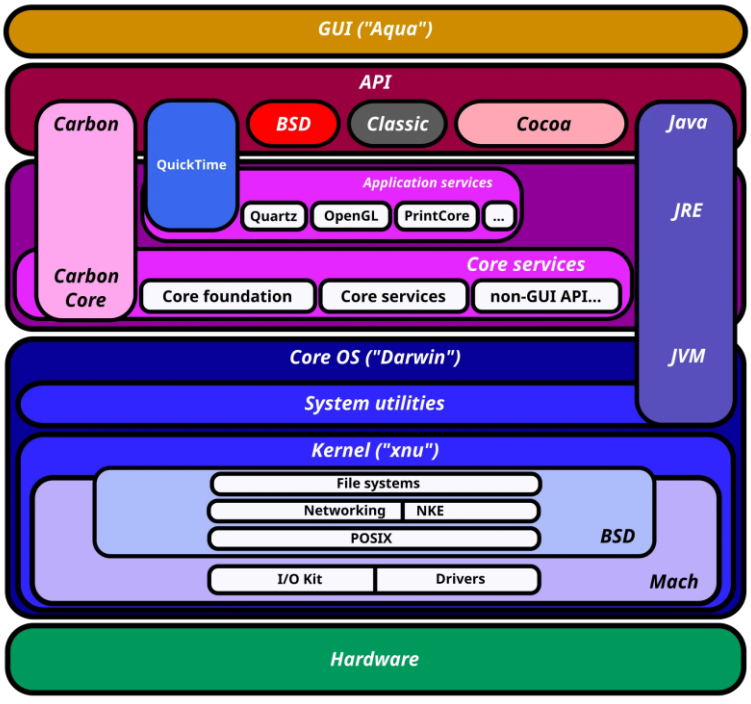
4.6. Extensões do Kernel (KEXTs)

O OS X trouxe um novo sistema de extensões para o kernel, o que possibilitou a adição de código ao núcleo do sistema de forma dinâmica, sem a necessidade de recompilar. Essas extensões, chamadas KEXTs, são utilizadas para a criação de drivers de dispositivos, compatibilidade com novos formatos de volume e suporte a protocolos de rede. Com essa abordagem, o sistema ganha mais modularidade e flexibilidade, tornando o desenvolvimento de software de baixo nível mais acessível e eficiente.

4.7. XNU

A documentação (APPLE, 2013) utilizada para a elaboração do conteúdo anterior sobre o kernel do macOS não nos deu informações sobre o XNU (X is Not Unix), que é realmente o Kernel propriamente dito do sistema operacional do macOS. O XNU é um kernel híbrido, pois combina o microkernel Mach com a camada BSD, que é monolitico.

Figura 4 - Arquitetura do macOS



Fonte: https://pt.wikipedia.org/wiki/Darwin_(sistema_operacional)

Como é possível ver nessa imagem, O XNU é quem vai fornecer o acesso ao BSD e ao Mach, garantindo a comunicação entre eles. O Darwin, é um projeto de código aberto que fornece algumas funções para o funcionamento do sistema.

O XNU, ao integrar o Mach e o BSD, garante a flexibilidade e modularidade do microkernel quanto os recursos avançados do Unix (APPLE, 2025).

5. SISTEMA DE ARQUIVOS

O desenvolvimento dos sistemas de arquivos da Apple começou com o HFS (Hierarchical File System), lançado em 1985 para substituir o sistema de arquivos do Macintosh original (MFS). O HFS trouxe uma abordagem hierárquica, organizando os dados em uma árvore de diretórios. No entanto, com o avanço da tecnologia, especialmente o surgimento de dispositivos de armazenamento como SSDs e a demanda por mais desempenho e segurança, o HFS tornou-se obsoleto.

Em 1998, a Apple introduziu o HFS+ (Mac OS Extended), que trouxe melhorias significativas, incluindo suporte a volumes maiores e mais eficiente utilização de armazenamento. No entanto, o HFS+ ainda possuía limitações, como a falta de suporte nativo para SSDs e a ineficiência em termos de segurança e desempenho para grandes volumes de dados. Além disso, o HFS+ não foi projetado para atender aos novos paradigmas computacionais, como a criptografia moderna e a gestão de grandes volumes de dados de maneira eficaz.

Com o avanço das tecnologias e a necessidade de um sistema de arquivos mais robusto e flexível, a Apple desenvolveu o Apple File System (APFS), lançado oficialmente em 2017. O APFS foi projetado para dispositivos modernos, incluindo SSDs, e incorpora tecnologias avançadas de segurança, como criptografia nativa, copy-on-write, e suporte a instantâneos (snapshots). O Apple File System (APFS), adotado como o sistema de arquivos padrão pelo macOS a partir da versão 10.13 (High Sierra), é um sistema de arquivos moderno e eficiente, projetado para otimizar o uso de armazenamento em SSDs e compatível com HDDs.

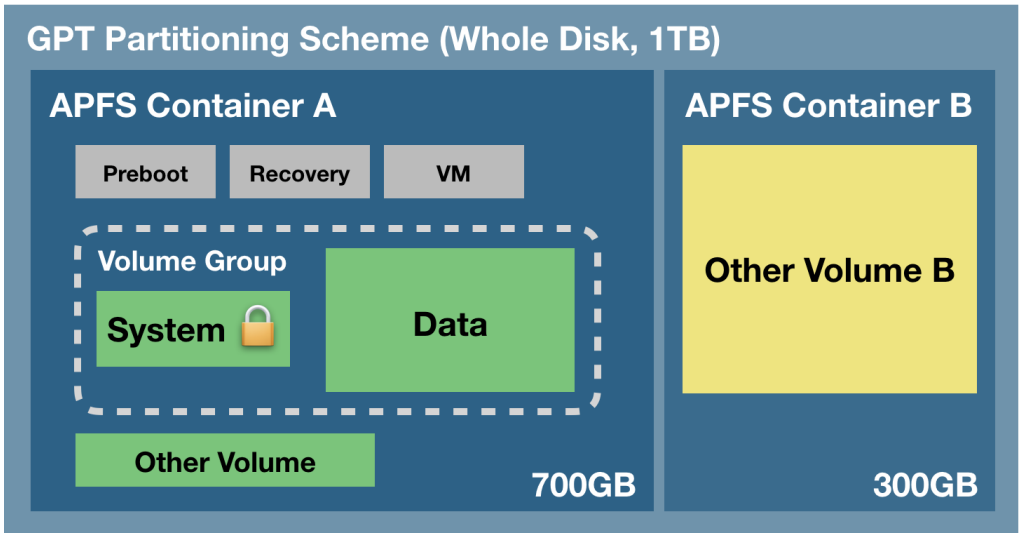
5.1 Estrutura do Apple File System (APFS)

O APFS introduz o conceito de contêineres e volumes, uma estrutura fundamental que melhora a flexibilidade e a gestão de armazenamento. Um contêiner no APFS pode conter múltiplos volumes, permitindo que diferentes volumes compartilhem o mesmo espaço de armazenamento físico. Os volumes são as divisões lógicas dentro de um contêiner, cada volume pode ser configurado de maneira independente e ter características próprias, como criptografia e capacidade de armazenamento. Para um volume, o tamanho mínimo exigido é de 512 MB. Para dois volumes, o tamanho mínimo aumenta para 1024 MB, garantindo que o sistema tenha espaço suficiente para gerenciar as alocações de dados de forma eficiente.

No APFS (Apple File System), a estrutura de armazenamento é gerida por meio de uma Tabela de Partição GUID (GPT), que organiza o espaço em diferentes volumes e recipientes. Os recipientes (containers) são unidades de armazenamento flexíveis que podem conter múltiplos volumes, com cada volume representando uma partição de dados.

O sistema inclui volumes ocultos, como o Volume de Pré-inicialização e o Volume de Recuperação, essenciais para inicialização e recuperação do sistema. Além disso, há volume de memória virtual (VM), responsável por armazenar arquivos temporários de troca, fundamental quando a memória RAM está saturada. Para maior organização e segurança, a partir do macOS Catalina, a Apple introduziu grupos de volume, separando o Volume do Sistema (protegido e imutável) do Volume de Dados (onde ficam armazenados arquivos de usuário e dados do sistema). Essa estrutura proporciona maior segurança, eficiência e flexibilidade na gestão de dados e recursos

Figura 5 – Estrutura da Tabela de Partição GUID (GPT)



Fonte: https://bombich.com/en/kb/ccc/5/working-apfs-volume-groups

5.1.1 Tipos de Blocos no APFS

O APFS usa diferentes tipos de blocos para gerenciar e armazenar dados. Cada tipo de bloco desempenha um papel essencial na estrutura do sistema de arquivos:

- Superbloco de contêiner (0x01): Contém informações sobre o contêiner, como o tipo de sistema de arquivos e metadados essenciais para sua operação.
- Nó (0x02): Representa arquivos e diretórios dentro do sistema de arquivos. Ele contém os metadados de um arquivo e suas referências para o conteúdo.
- Gerente de espaço (0x05): Gerencia a alocação de espaço no disco, garantindo que os dados sejam armazenados de maneira eficiente e sem fragmentação excessiva.
- Arquivo de informações de alocação (0x07): Armazena informações sobre os blocos de dados alocados, permitindo o gerenciamento da alocação e liberação de espaço.
- Árvore B (0x0B): Estrutura de dados usada para organizar e pesquisar eficientemente arquivos e diretórios. A árvore B oferece um desempenho rápido ao localizar dados.
- Ponto de verificação (0x0C): Representa um ponto específico de recuperação do sistema de arquivos. Ele é usado para garantir a integridade do sistema em caso de falha.

- Superbloco de volume (0x0D): Contém informações cruciais sobre o volume, como seu tamanho, estrutura e status de criptografia.

5.1.2 Cabeçalho do Bloco

Cada bloco no APFS possui um cabeçalho que contém informações essenciais sobre o bloco:

Tabela 1 - Cabeçalho do bloco

Soma de verificação	Garantia de integridade do bloco, permitindo detectar corrupção de dados.
ID do bloco	Identificador único do bloco no sistema de arquivos.
Versão	Indica a versão do sistema de arquivos ou do tipo de bloco.
Tipo de bloco	Define o tipo de dados contidos no bloco, como superbloco, nó ou árvore B.
Flags	Flags que indicam o estado ou características especiais do bloco.
Preenchimento	Áreas de dados não usadas, que podem ser utilizadas para alinhamento ou dados futuros.

5.2 Funcionalidades do APFS

5.2.1 Copy-on-Write (COW)

O copy-on-write (COW) é um mecanismo fundamental do APFS que melhora a segurança e a eficiência do sistema de arquivos. Ao invés de sobrescrever dados existentes, o sistema cria uma nova cópia dos dados modificados. Isso garante que os dados anteriores permaneçam intactos, permitindo a recuperação de estados anteriores do sistema de arquivos. O COW também ajuda a evitar corrupção de dados em caso de falha, como uma queda de energia.

5.2.2 Instantâneos (Snapshots)

O APFS introduz a capacidade de criar snapshots, ou "instantâneos", que capturam o estado do sistema de arquivos em um determinado momento. Isso permite

uma forma eficiente de backup, já que o snapshot rastrea apenas as alterações feitas após a criação, ao invés de criar cópias completas de arquivos. Esse recurso é amplamente utilizado no Time Machine, o sistema de backup da Apple.

5.2.3 Criptografia Nativa

O APFS oferece criptografia nativa com dois esquemas principais: AES-XTS e AES-CBC, que garantem a proteção de dados no nível do sistema de arquivos. Além disso, o ApFS suporta criptografia multichave, permitindo que arquivos e metadados sejam criptografados com chaves diferentes, proporcionando uma segurança mais robusta.

5.2.4 Suporte a TRIM

O comando TRIM ajuda a gerenciar blocos obsoletos em SSDs. O APFS executa operações TRIM de forma assíncrona após a gravação dos metadados em mídias estáveis, o que melhora o desempenho e prolonga a vida útil dos dispositivos de armazenamento flash.

5.2.5 Precisão de Nanossegundos

O APFS utiliza carimbos de data/hora com precisão de nanossegundos, em contraste com a precisão de segundos do HFS+. Essa precisão adicional é crucial para aplicações que exigem registros e agrupamentos de eventos altamente detalhados.

5.3 Metodologias de Recuperação de Dados

Os pontos de verificação no APFS permitem recuperar o estado do sistema de arquivos em momentos específicos. Eles são essenciais para a integridade dos dados e podem ser usados para restaurar versões anteriores dos arquivos. A cadeia de superblocos de contêiner facilita a recuperação de versões mais antigas, assegurando que a integridade dos dados seja mantida mesmo após falhas.

O carving é o processo de recuperação de dados de espaços não alocados. No APFS, existem três métodos principais de carving:

- Escultura NXSB: Foca no superbloco de contêiner.
- Escultura APSB: Foca no superbloco de volume.

- Escultura de Nó: Foca em nós de arquivo.

Carving no APFS apresenta desafios devido à ausência de limites rígidos entre volumes, o que torna a identificação de dados fragmentados mais complexa.

6. GERENCIAMENTO DE MEMÓRIA

O macOS é projetado para gerenciar a memória de maneira altamente eficiente, combinando tecnologia de hardware avançada com algoritmos de software sofisticados. O gerenciamento de memória visa garantir desempenho superior e estabilidade, mesmo durante períodos de alta carga de trabalho. O sistema de gerenciamento de memória do macOS é robusto e dinâmico, garantindo que o uso de recursos seja otimizado para as diversas necessidades de aplicações e usuários. A seguir, exploramos os principais aspectos do gerenciamento de memória no macOS.

6.1 Memória Virtual e Volume de VM

A memória virtual no macOS permite que o sistema operacional expanda a capacidade da memória RAM física ao utilizar o disco rígido ou SSD como uma extensão da memória. Isso é particularmente útil quando a memória física disponível não é suficiente para atender às necessidades dos processos ativos. No macOS, essa funcionalidade é suportada pelo Volume de VM (Virtual Memory), que é um volume dedicado dentro do sistema de arquivos APFS.

O Volume de VM armazena arquivos temporários de troca (swap), que contêm dados de processos que não estão sendo ativamente usados na RAM. Quando a memória física está saturada, o macOS pode transferir partes dos dados menos acessados para o disco, liberando espaço na RAM para novos dados de processos prioritários. Esse processo de troca é transparente para o usuário, mas é fundamental para a manutenção da estabilidade do sistema.

No APFS, o Volume de VM é projetado para ser altamente eficiente em termos de leitura e escrita. Como o APFS gerencia blocos de dados e metadados de maneira eficiente, a latência durante a leitura e escrita dos arquivos de paginação (swap files) é reduzida significativamente. Isso ajuda a manter o desempenho do sistema, mesmo sob carga elevada, quando há uma quantidade significativa de dados sendo movida

entre a RAM e o disco. A eficiência do gerenciamento do Volume de VM no APFS também contribui para uma melhor segurança, já que as operações de swap podem ser feitas de maneira rápida e segura.

6.2 Alocação Dinâmica e Compressão

O macOS implementa técnicas avançadas para otimizar o uso da memória disponível, uma das quais é a compressão de memória. Quando a RAM está se tornando escassa, o sistema pode compactar dados que não estão sendo ativamente utilizados. Isso reduz a quantidade de memória necessária para armazenar esses dados, liberando espaço para aplicativos e processos prioritários.

Essa compressão de memória acontece de forma transparente e em tempo real. O sistema comprime os dados não utilizados e os mantém em uma parte compactada da memória, permitindo que a RAM seja alocada de maneira mais eficiente para as tarefas em andamento. O processo é muito eficiente, garantindo que os dados comprimidos possam ser descompactados rapidamente quando necessário, sem comprometer significativamente o desempenho.

Além disso, o macOS também faz uso de alocação dinâmica de memória, ajustando a distribuição de memória conforme a demanda. Isso significa que o sistema aloca memória conforme os processos a solicitam, priorizando aqueles que estão em uso ativo. Quando um processo se torna o foco do usuário ou do sistema, o macOS aloca mais memória a ele, garantindo que o desempenho seja mantido para as aplicações mais importantes, ao mesmo tempo em que libera memória de processos menos prioritários.

6.3 Proteção e Isolamento

Para garantir a segurança e a estabilidade do sistema, o macOS implementa mecanismos de proteção e isolamento de memória entre os diferentes processos em execução. Cada processo no macOS tem seu próprio espaço de endereçamento de memória, o que significa que ele é isolado de outros processos e não pode acessar a memória alocada para outro processo sem permissões explícitas. Esse isolamento é essencial para a segurança do sistema, pois impede que processos maliciosos ou

com falhas acessem dados críticos ou corrompam o funcionamento de outros processos. Além disso, o macOS adota várias tecnologias de segurança para proteger a memória do sistema contra ataques e exploits. Duas dessas tecnologias são:

Execute Disable (XD): Essa tecnologia impede que código seja executado em áreas de memória onde não deveria ser, como áreas de pilha ou de heap, protegendo contra buffer overflows e executação de código malicioso. Com o XD, mesmo que um atacante consiga injetar código malicioso em uma área da memória, ele não poderá executá-lo.

Address Space Layout Randomization (ASLR): A ASLR randomiza a localização de importantes áreas de memória, como a pilha de execução e o heap, dificultando os ataques que dependem de conhecimento de onde certos dados ou funções estão localizados na memória. Isso torna muito mais difícil para os atacantes preverem ou manipularem a estrutura de memória do sistema, protegendo o sistema contra uma ampla gama de exploits.

7 GERENCIAMENTO DE PROCESSOS

7.1 Como o Mac OS lida com os processos

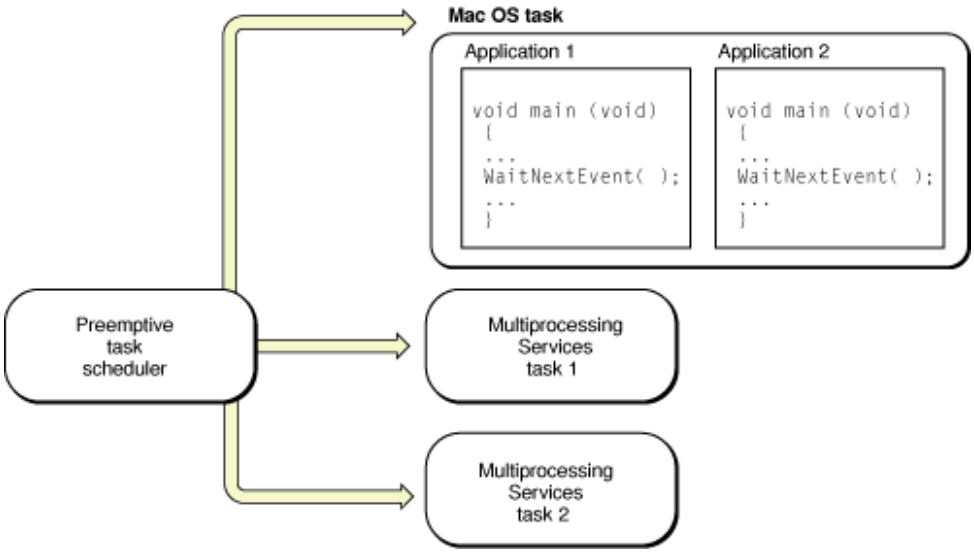
Os seguintes tópicos abordarão a forma como o Mac OS realiza o gerenciamento de processos, além de citar os principais métodos de comunicação entre processos (IPC) utilizados pelo sistema operacional. Ao fim, também será apresentada uma breve explicação sobre elementos do terminal do Mac OS, referentes a processos e uso da CPU.

7.1.1 Escalonamento de processos

O escalonamento de processos no Mac OS combina duas abordagens: a cooperativa e a preemptiva. No escalonamento cooperativo, uma tarefa mantém o controle do processador até que voluntariamente o libere, permitindo a execução de outras tarefas. Já no escalonamento preemptivo, o sistema operacional pode interromper uma tarefa em execução para dar prioridade a outra. Especificamente no Mac OS, uma aplicação que utiliza escalonamento cooperativo pode criar tarefas que serão executadas de forma preemptiva. Porém, a tarefa principal da aplicação(que contém o loop de eventos) não é interrompida diretamente pelo sistema. Em vez disso, ela opera dentro de uma tarefa gerenciada pelo Mac OS, que é escalonada de modo

preemptivo. Assim, dentro da estrutura, a aplicação deve compartilhar o processador cooperativamente com outras aplicações em execução.

Figura 6



Fonte: https://developer.apple.com/library/archive/documentation/Carbon/Conceptual/Multitasking_MultiproServ/02concepts/concepts.html

Uma tarefa permanece em execução até que seja concluída, bloqueada ou interrompida. O bloqueio ocorre quando a tarefa aguarda um evento, sendo removida da fila até que o evento ocorra ou o tempo limite seja atingido. Caso não finalize ou seja bloqueada dentro do tempo estipulado, o escalonador a interrompe, movendo-a para o final da fila e permitindo a execução de outra tarefa. Caso a tarefa principal seja bloqueada ao aguardar um evento dos Multiprocessing Services, a aplicação não retorna ao loop de eventos até que ele ocorra, podendo comprometer a execução. Assim, é normalmente utilizada a verificação periódica (polling) de eventos no loop, evitando bloqueios desnecessários.

7.1.2 Arquiteturas de Tarefas

A divisão do trabalho em tarefas depende do tipo de processamento e da interdependência entre elas. Em sistemas com multiprocessadores, é essencial otimizar a multitarefa para manter os processadores ocupados. Isso pode ser feito criando várias tarefas e permitindo que o escalonador as distribua automaticamente ou ajustando o número de tarefas conforme a quantidade de processadores disponíveis. O programa pode monitorar essas tarefas no loop de eventos até a conclusão do processamento.

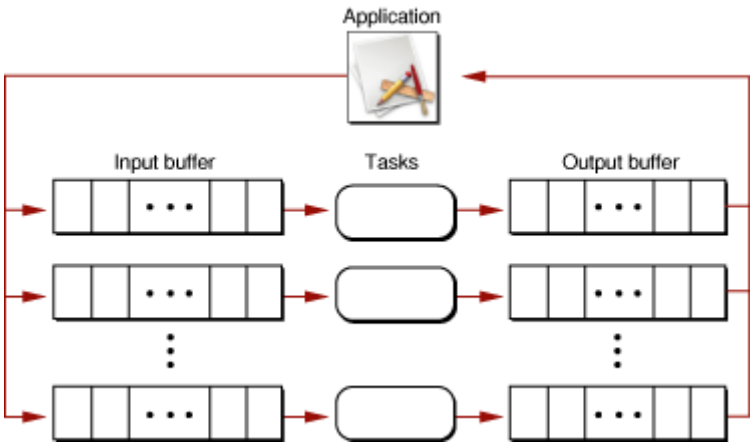
7.1.3 Múltiplas Tarefas Independentes

Em muitos casos, é possível dividir os aplicativos em diferentes seções que não necessariamente dependem umas das outras, mas que idealmente seriam executadas simultaneamente. É possível exemplificar isso com um aplicativo que consiga ter uma seção de código para renderizar imagens na tela, outra para fazer cálculos gráficos em segundo plano e uma terceira para baixar dados de um servidor. Cada uma dessas seções é candidata natural para tarefas preemptivas. Mesmo que apenas um processador esteja disponível, geralmente é vantajoso ter essas seções independentes executadas como tarefas preemptivas. O aplicativo pode notificar as tarefas (usando qualquer um dos três mecanismos de notificação) e, em seguida, pesquisar os resultados dentro de seu loop de eventos.

7.1.4 Tarefas Paralelas com Buffers de E/S Paralelos

Quando o trabalho computacional de um aplicativo pode ser dividido em partes semelhantes, pode-se criar tantas tarefas quanto processadores disponíveis e distribuir o trabalho uniformemente entre eles. Por exemplo, uma tarefa de filtragem em uma imagem pode ser dividida em seções iguais, com cada processador responsável por uma fração do trabalho. Para isso, é necessário criar dois buffers por tarefa: um para receber “solicitações de trabalho” e outro para postar os resultados. As tarefas solicitam trabalho do buffer de entrada, executam a tarefa e postam os resultados no buffer de saída.

Figura 7



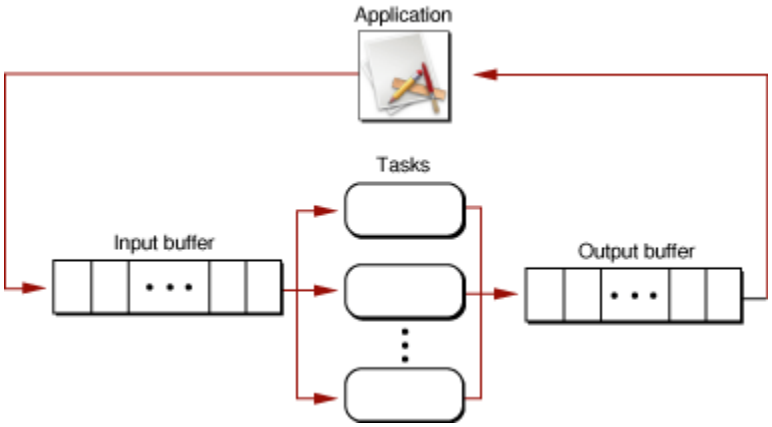
Fonte: https://developer.apple.com/library/archive/documentation/Carbon/Conceptual/Multitasking_MultiproServ/02concepts/concepts.html

7.1.5 Tarefas Paralelas com um Conjunto Único de Buffers de E/S

Se o tempo de execução das tarefas for imprevisível, pode-se usar um único buffer de entrada compartilhado por todas as tarefas. O aplicativo coloca as

solicitações de trabalho nesse buffer, e as tarefas solicitam uma tarefa quando estiverem disponíveis. Após processar, a tarefa coloca o resultado em um único buffer de saída. O uso de um buffer de entrada único permite que o trabalho seja distribuído de forma dinâmica, mas torna difícil prever a ordem de processamento ou os resultados.

Figura 8

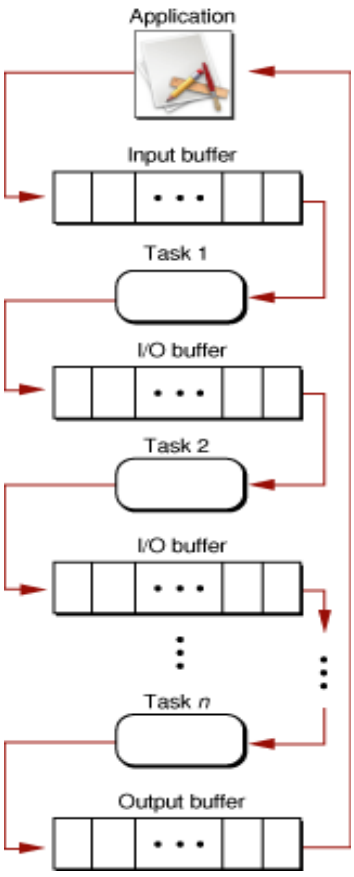


Fonte: https://developer.apple.com/library/archive/documentation/Carbon/Conceptual/Multitasking_MultiproServ/02concepts/concepts.html

7.1.6 Tarefas Sequenciais

Em algumas aplicações, as tarefas devem ser realizadas em sequência. Pode-se exemplificar com o caso de um aplicativo de efeitos sonoros que processa diferentes efeitos de forma ordenada. Nesse caso, o resultado de uma tarefa serve de entrada para a próxima, formando uma arquitetura sequencial ou "pipeline". Embora a arquitetura sequencial aproveite múltiplos processadores apenas se as tarefas puderem operar simultaneamente, pode-se adicionar tarefas paralelas em etapas individuais do pipeline para otimizar o uso dos processadores.

Figura 9

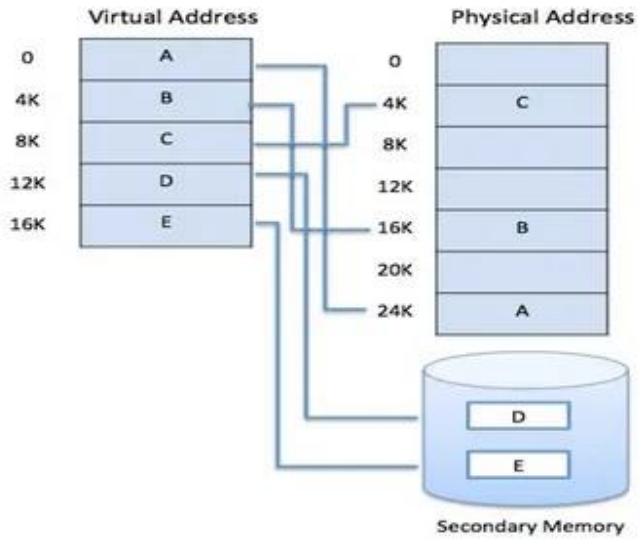


Fonte: https://developer.apple.com/library/archive/documentation/Carbon/Conceptual/Multitasking_MultiproServ/02concepts/concepts.html

7.1.7 Memória virtual

A memória virtual é uma conjunção da RAM e do hardware de disco. Quando a RAM está toda preenchida com os processos, mas ainda precisa de memória extra, o Mac OS utiliza espaço de armazenamento do disco, transferindo temporariamente alguns dos processos relativamente menos ativos para o disco de hardware, liberando espaço na RAM para processos de maior prioridade. Ao usar a memória virtual, o sistema pode executar um programa que é maior do que sua memória física, mas os processos que estão no disco de hardware precisam de mais tempo para serem executados.
Por exemplo, um programa de 2 Megabytes pode ser executado em uma máquina de 256 Kilobytes, agendando cuidadosamente a escolha de quais 256 KB serão executados a cada instante.

Figura 10



Fonte: https://medium.com/@qiming.wei/macos-resource-management-81ad9144ab6

7.2 Comunicação interprocessos (IPC)

A comunicação entre processos (IPC - Interprocess Communication) refere-se às técnicas para troca de dados entre processos e threads. As técnicas IPC comuns no Mac OS X incluem memória compartilhada, portas Mach, soquetes, notificações distribuídas, áreas de transferência e objetos distribuídos. Também será analisado o método IPC de Apple Events, próprio do sistema Mac OS.

7.2.1 Memória compartilhada

É uma implementação para IPC onde uma região da memória é compartilhada entre diferentes processos. A exemplo disso, um processo A poderia escrever nessa região da memória e B poderia ler desta memória, ou vice-versa. Isto permite uma rápida comunicação e os dados não precisam ser copiados. Entretanto, apresenta desvantagem em relação à dificuldade de coordenar as mudanças na área de memória compartilhada.

7.2.2 Mach Ports

Mach Ports são uma estrutura para comunicação entre processos (IPC) no Mach Kernel. Elas funcionam como filas estruturadas de mensagens gerenciadas pelo kernel, permitindo que processos e o sistema operacional troquem informações. Cada porta possui “direitos de acesso”(port right), que são identificadores numéricos que definem se um processo pode enviar ou receber mensagens. Os tipos de Port Right são: Receive Right(permite receber mensagens), Send Right(permite enviar mensagens para uma porta), Send-Once Right(permite enviar uma única mensagem e depois expira), Port Set Right(agrupa várias portas, permitindo monitorar múltiplas portas simultaneamente) e Dead Name(representa uma porta destruída).

7.2.2 Sockets

Sockets podem ser utilizados na comunicação entre processos (IPC) tanto localmente quanto por rede, permitindo uma interação no próprio computador ou entre computadores diferentes. Os sockets funcionam como uma porta de comunicação, permitindo que os programas enviem e recebam dados por meio deles, como mensagens, arquivos ou comandos. Desse modo, quando um cliente cria um Socket, um servidor pode se conectar através de um endereço e porta, possibilitando a troca de dados entre eles.

7.2.3 Pasteboard

Toda vez que um copiar-colar acontece entre aplicativos, ocorre um IPC por meio de pasteboard, assim como em operações de arrastar e soltar entre aplicativos. É possível criar pasteboards personalizados que somente os aplicativos desejados podem acessar para passar dados de um lado para o outro entre aplicativos. Seu funcionamento se deve à comunicação com um servidor central de pasteboard, que utiliza Mach Ports.

7.2.4 Distributed Objects

Esse mecanismo permite que um processo chame um objeto em um processo diferente (ou um thread diferente no mesmo processo). Os processos podem até mesmo estar rodando em computadores diferentes em uma rede. Distributed Objects operam fazendo com que o servidor torne público um objeto ao qual outros processos clientes podem se conectar. Uma vez que uma conexão é feita, o processo cliente pode chamar um dos métodos do objeto público, como se o objeto existisse no processo cliente. Normalmente esse método é executado em Mach Ports, mas também podem ser usados com sockets, permitindo que funcionem entre computadores.

7.3.1 Apple Events (método de comunicação entre processos)

Apple Events é um dos métodos de comunicação entre processos (IPC) utilizado no Mac OS, especialmente por aplicativos de interface gráfica (GUI). Ele possibilita a interação entre aplicativos, realizando operações como abrir arquivos, enviar comandos ou encerrar processos. Os Apple Events podem encapsular comandos e dados complexos, além de fornecerem um mecanismo de transporte de dados e despacho de eventos, podendo ser usados em um único aplicativo, entre aplicativos no mesmo computador e entre aplicativos em diferentes computadores, conectados a uma rede.

Os Apple Events são parte da Open Scripting Architecture (OSA), que padroniza e expande a comunicação entre aplicativos no Mac OS X.

7.3.2 AppleScript:

É uma linguagem de programação desenvolvida pela Apple, que permite a criação de scripts para realizar envios de Apple Events, possibilitando automatizar tarefas e instruir aplicativos, por meio de eventos, a realizar determinadas operações. Por exemplo, um script pode ser criado para abrir um documento em um editor de texto, inserir texto e salvar o arquivo.

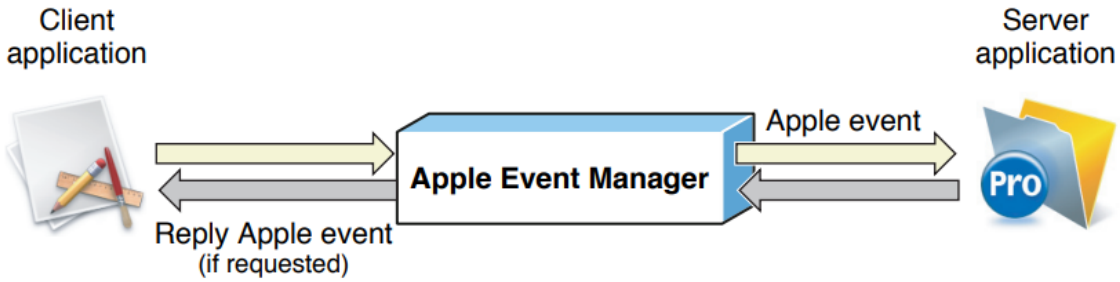
7.3.3 Como Apple Events funcionam:

Um Apple Event especifica um aplicativo de destino (ou outro processo) e fornece uma descrição detalhada de uma operação a ser executada. O sistema operacional localiza o destino, entrega o evento e, se necessário, retorna outro Apple Event como resposta ao remetente. Esse mecanismo é útil para interação entre processos e para automatizar tarefas que usam vários aplicativos, sendo frequentemente utilizados para iniciar e encerrar aplicativos ou para trocar informações e comandos entre eles.

7.3.4 Enviando e recebendo Apple Events:

Os aplicativos normalmente usam Apple Events para solicitar serviços e dados de outros aplicativos ou para fornecer serviços e eventos em resposta a tais solicitações. Em termos cliente-servidor, o aplicativo cliente
envia um Apple Event para solicitar um serviço ou informação do aplicativo servidor. A Figura exemplifica dois aplicativos se comunicando, com o cliente usando funções do Apple Event Manager para interagir com o servidor.

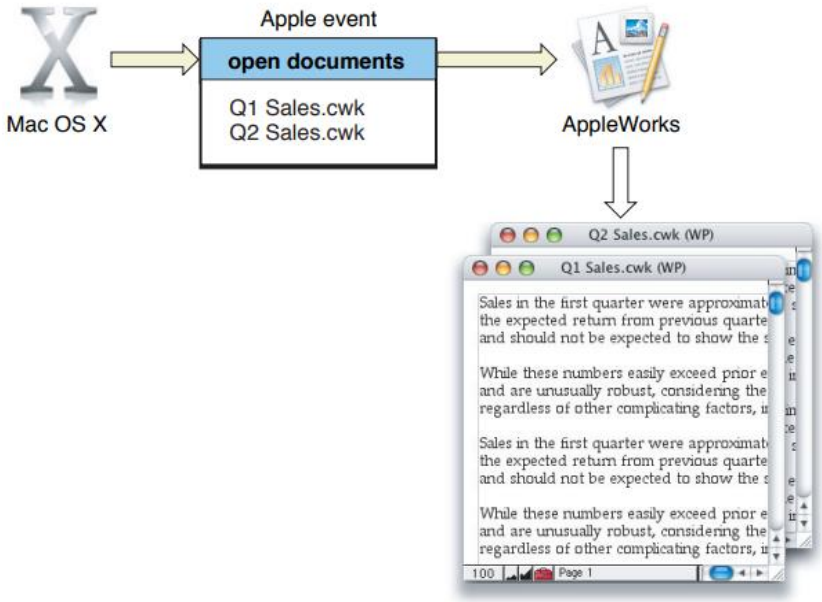
Figura 11



Fonte: https://applescriptlibrary.wordpress.com/wp-content/uploads/2013/11/apple-events-programming-guide.pdf

Aplicativos também devem estar preparados para responder a Apple Events enviados pelo Mac OS, bem como a outros eventos que o aplicativo suporta. Os Apple Events normalmente contém informações que especificam o aplicativo de destino e a ação a ser executada. A Figura a seguir exemplifica um Apple Event enviado pelo Mac OS X para o aplicativo AppleWorks. Esse tipo de evento fornece uma lista de arquivos para o aplicativo de destino abrir.

Figura 12



Fonte: https://applescriptlibrary.wordpress.com/wp-content/uploads/2013/11/apple-events-programming-guide.pdf

7.3.5 Quando os aplicativos usam Apple Events:

- Para responder aos Apple Events recebidos do Mac OS.

Para aplicativos que apresentam uma interface gráfica de usuário, o Mac OS X envia eventos para iniciar certas operações, como iniciar ou encerrar aplicativos.

- Para disponibilizar seus dados e serviços a outros processos.

A maioria dos aplicativos que fornecem serviços por meio de Apple Events são aplicativos programáveis. Eles fornecem uma terminologia de script que permite que os usuários escrevam instruções em linguagem AppleScript para acessar as operações e dados do aplicativo. Quando um script é executado, algumas de suas instruções resultam em Apple Events sendo enviados ao aplicativo.

- Para se comunicar diretamente com outros aplicativos.

Um aplicativo pode enviar um Apple Event para solicitar que outro aplicativo execute uma operação ou retorne dados.

- Para oferecer suporte à gravação em um aplicativo programável.

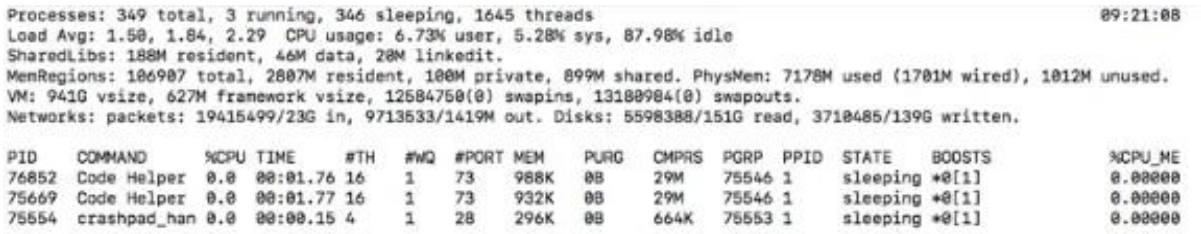
Gravações permitem que ações executadas manualmente por um usuário sejam registradas em um script. Ou seja, os usuários podem ativar a gravação no aplicativo Script Editor e, em seguida, executar ações com aplicativos que suportam gravações.

Quaisquer Apple Events gerados por essas ações são registrados em um script em AppleScript.

7.4 Comando TOP no terminal do Mac OS:

O comando “TOP” no terminal do Mac OS exibe todas as tarefas gerenciadas pelo kernel do computador em tempo real.

Figura 13



Fonte: https://medium.com/@qiming.wei/macos-resource-management-81ad9144ab6

1. A primeira linha apresenta a situação dos processos. No exemplo, há 349 processos no total, sendo que 3 estão em execução e 346 estão dormindo. Ademais, também é exibido o número de threads totais em todos os programas (1645 threads).

2. Na segunda linha, “CPU usage” exibe a porcentagem de uso do processador, dividida entre usuários(user), sistema(sys) e componentes ociosos(idle).

3. Na terceira linha, “SharedLibs” refere-se às bibliotecas compartilhadas que são usadas por vários processos. No exemplo, "188M resident" indica a quantidade de memória utilizada por essas bibliotecas.

4. Na quinta linha, “VM” exibe o tamanho total da sua memória virtual, incluindo os números, swapins e swapouts.

5. Na sexta linha, “Networks packets” exibe a quantidade de pacotes de dados que foram processados pela rede. No exemplo, "194156979/236 in" representa o número de pacotes recebidos e "9713533/1419M out" representa o número de pacotes enviados.

6. A coluna PID mostra o ID de cada processo.

7. A coluna “COMMAND” exibe o nome de cada processo.

8. A coluna “%CPU” exibe o uso da CPU de cada processo específico.

9. A coluna “#PORT” exibe o número de portas de comunicação(Mach Ports) que um processo está utilizando.

10. A coluna “MEM” exibe o tamanho da memória interna de cada processo.

11. A coluna “PURG” exibe a quantidade de memória que um processo tem em estado "purgable", indicando que pode ser liberada do sistema.

12. A coluna “CMPRS” exibe quanto de memória de um processo foi comprimida pelo sistema para otimizar o uso da RAM.

13. A coluna “PGRP” exibe a qual grupo de processos o qual um processo pertence. Isso é útil para gerenciar e controlar grupos de processos relacionados.

14. A coluna “PGRP” exibe o ID do grupo de processos ao qual um processo pertence.

15. A coluna “PPID” exibe o ID do processo pai que criou o processo atual.

16. A coluna “STATE” exibe o estado atual de cada processo.

8 CONSIDERAÇÕES FINAIS

Por fim, é possível concluir que o macOS é um sistema operacional altamente consolidado, isso porque tem uma estrutura altamente confiável e eficiente. A análise feita sobre o Kernel, o sistema de arquivo, gerenciamento de processos, gerenciamento de memória, e os demais tópicos revelam tamanha robustez e sofisticação do sistema, garantindo grande fluidez e confiabilidade no uso. Ademais, as constantes atualizações e updates mostram o compromisso da Apple em garantir um sistema sempre modernizado e relevante no mercado de tecnologia mundial.

Portanto, esse estudo colabora para uma melhor compreensão de como o macOS funciona, disponibilizando ponto de vista mais detalhado sobre seus componentes, sua estrutura e suas vantagens.

9 REFERÊNCIAS BIBLIOGRÁFICAS

APPLE INC. Darwin XNU. *GitHub*, 2025. Disponível em: https://github.com/apple/darwin-xnu. Acesso em: 17 fev. 2025.

APPLE INC. Kernel Architecture Overview. *Apple Developer Documentation*, 2013. Disponível em: https://developer.apple.com/library/archive/documentation/Darwin/Conceptual/KernelProgramming/Architecture/Architecture.html. Acesso em: 17 fev. 2025.

APPLE INC. Monitor de Atividade: Como usar o Monitor de Atividade no macOS. Disponível em: https://support.apple.com/pt-br/guide/activity-monitor/actmntr1004/mac. Acesso em: 16 fev. 2025.

APPLE INC. Utilitário de Disco: Como usar o Utilitário de Disco no macOS. Disponível em: https://support.apple.com/pt-br/guide/disk-utility/dsku19ed921c/mac. Acesso em: 16 fev. 2025.

BOMBICH SOFTWARE. *Working with APFS Volume Groups*. Disponível em: https://bombich.com/en/kb/ccc/5/working-apfs-volume-groups. Acesso em: 17 fev. 2025.

KINGSTON TECHNOLOGY COMPANY, INC. Understanding File Systems. Disponível em: https://www.kingston.com/br/blog/personal-storage/understanding-file-systems. Acesso em: 16 fev. 2025.

NTFS.COM. *Benefits of APFS*. Disponível em: http://ntfs.com/apfs-benefits.htm. Acesso em: 17 fev. 2025.

WONDERSHARE. APFS: o novo sistema de arquivos da Apple. Disponível em: https://recoverit.wondershare.com.br/mac-data-recovery/APFS-new-apple-file-system.html. Acesso em: 16 fev. 2025.

WONDERSHARE. What is HFS File System? Disponível em: https://recoverit.wondershare.com.br/file-system/what-is-hfs-file-system.html. Acesso em: 16 fev. 2025.

APPLE-HISTORY. *Apple-History.com*. Disponível em: http://www.apple-history.com/. Acesso em: 17 fev. 2025.

APPLE INC. *File System Documentation*. Disponível em: https://developer.apple.com/documentation/foundation/file_system?language=objc. Acesso em: 17 fev. 2025.

APPLE INC. *Apple File System Reference*. Disponível em: https://developer.apple.com/support/downloads/Apple-File-System-Reference.pdf. Acesso em: 17 fev. 2025.

WEI, Qiming. *macOS Resource Management*. Medium, 2019. Disponível em: https://medium.com/@qiming.wei/macos-resource-management-81ad9144ab6. Acesso em: 17 fev. 2025.

SLIDESHARE. *IPC on Mac OS X*. Disponível em: https://pt.slideshare.net/slideshow/ipc-on-mac-osx/39649434. Acesso em: 17 fev. 2025.

APPLE INC. *Apple Events Programming Guide*. Disponível em: https://applescriptlibrary.wordpress.com/wp-content/uploads/2013/11/apple-events-programming-guide.pdf. Acesso em: 17 fev. 2025.

APPLE INC. *Multitasking and Multiple Process Support*. Disponível em: https://developer.apple.com/library/archive/documentation/Carbon/Conceptual/Multitasking_MultiproServ/02concepts/concepts.html. Acesso em: 17 fev. 2025.

DARLINGHQ. *Mach Ports in macOS*. Disponível em: https://docs.darlinghq.org/internals/macos-specifics/mach-ports.html. Acesso em: 17 fev. 2025.