Mac OS X

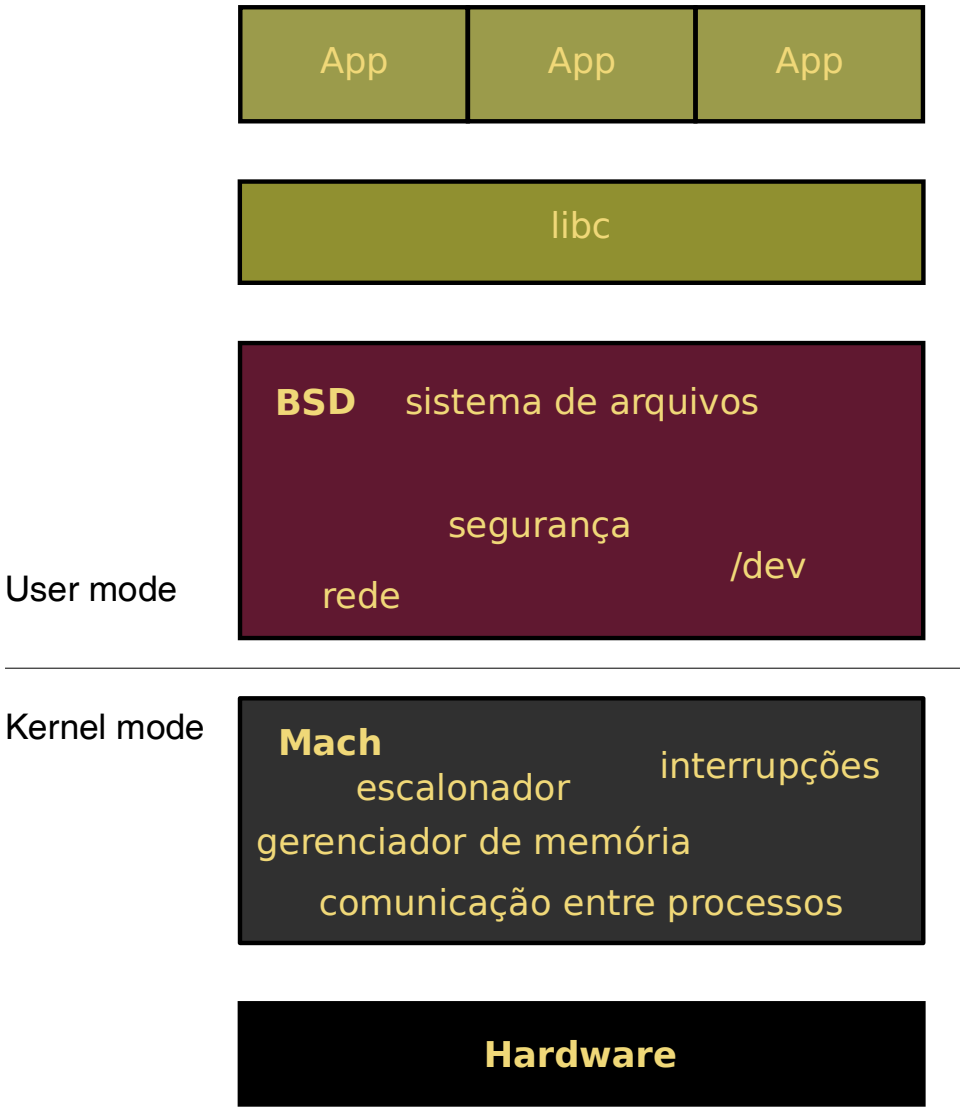
Felipe Gomes Lacerda Pedro Garcia Freitas

23 de novembro de 2009

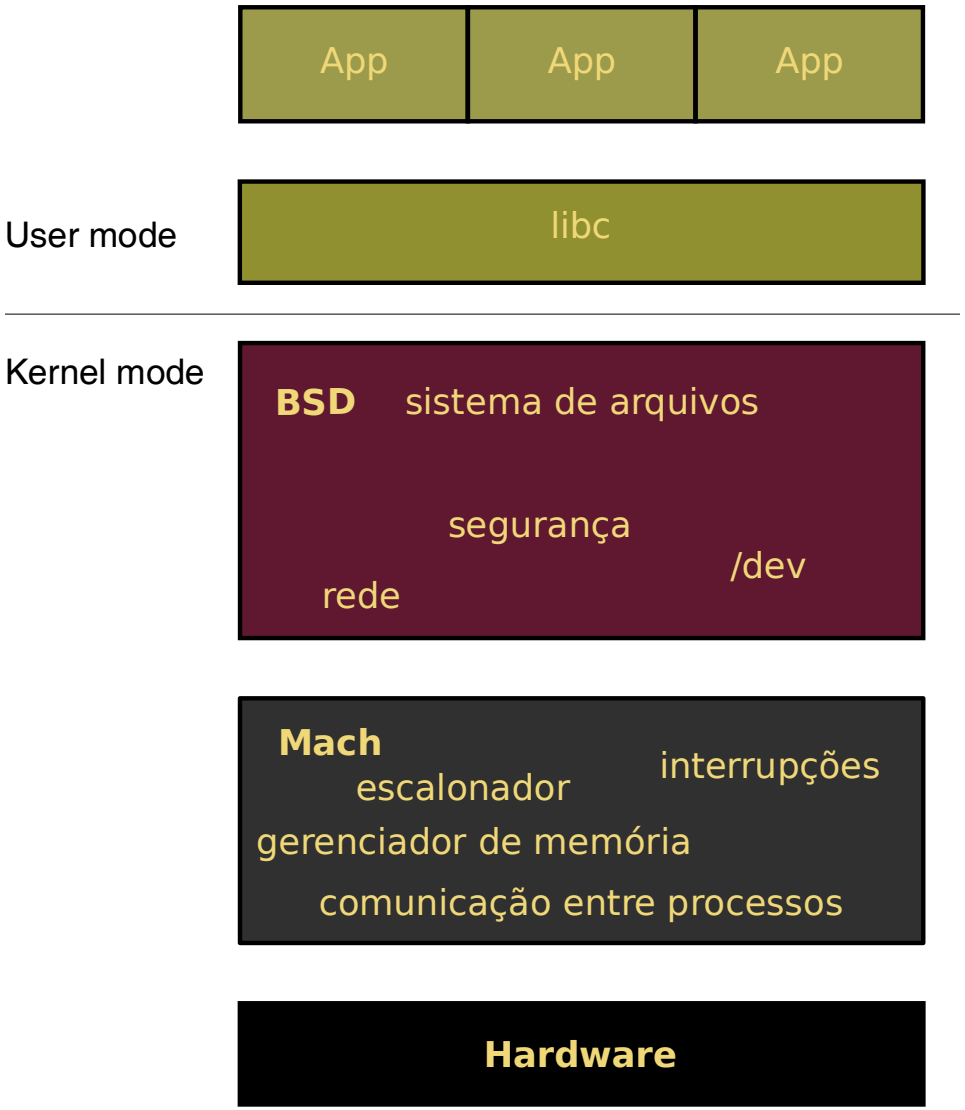
Modelo do kernel

- Microkernel:
  - Uso mínimo de espaço de kernel
  - Fornece apenas os mecanismos básicos; as *políticas* ficam a cargo do usuário
  - Kernel: IPC "baixo nível", gerenciamento de memória, escalonamento, tratamento de interrupções
  - Usuário: sistema de arquivos, drivers, rede, segurança, IPC "alto nível"
- Mac OS X:
  - Camadas: BSD e Mach
  - BSD: camada Unix-like
    - Roda em modo kernel (apenas por razões de eficiência)
  - Mach: serviços básicos de um microkernel

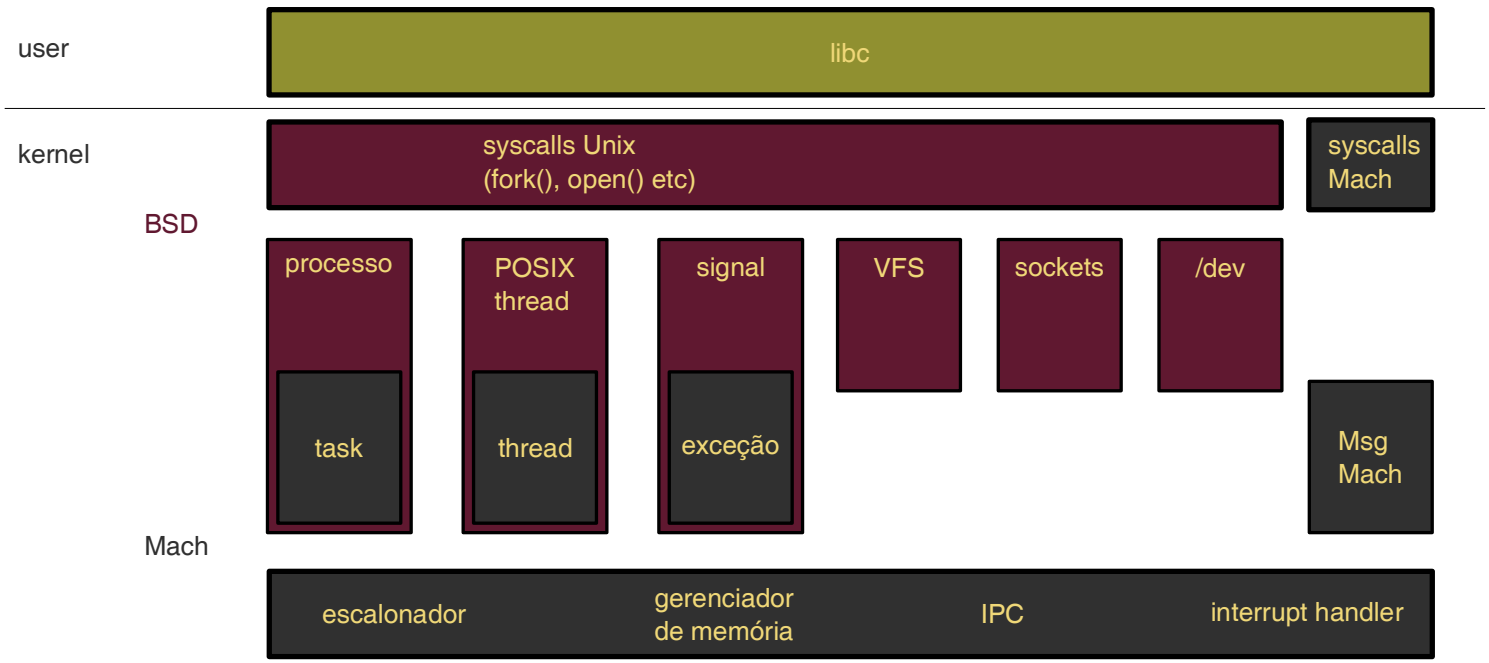
Estrutura


App
App
App
libc
BSD
sistema de arquivos
segurança
/dev
rede
User mode
Kernel mode
Mach
escalonador
interrupções
gerenciador de memória
comunicação entre processos
Hardware

Estrutura


App
App
App
User mode
libc
Kernel mode
BSD
sistema de arquivos
segurança
/dev
rede
Mach
escalonador
interrupções
gerenciador de memória
comunicação entre processos
Hardware

Estrutura


user
libc
kernel
syscalls Unix
(fork(), open() etc)
syscalls
Mach
BSD
processo
task
POSIX
thread
thread
signal
exceção
VFS
sockets
/dev
Msg
Mach
Mach
escalonador
gerenciador
de memória
IPC
interrupt handler

Processos, tasks e threads

- Processo: PID, arquivos abertos, lista de pai/filhos, credenciais do usuário, limites do processo, e a *task* correspondente
- Task: espaço de endereçamento, lista de threads, prioridade base e máxima, estruturas para IPC, e o endereço do processo correspondente
- Thread: prioridade, fila de execução, fila de espera, estatísticas de uso do processador, estado, estruturas para IPC

Escalonador

- Escalona threads, não processos
- Algoritmo: filas múltiplas com prioridade (uma fila para cada prioridade)

Banda de prioridade	Características	Faixa
Normal	prioridade padrão para aplicações de nível de usuário	0-63
Prioridade alta de sistema	threads do sistema que não estão na camada Mach	64-79
Modo kernel	threads criadas dentro do kernel que precisam executar a uma prioridade mais alta	80-95
Tempo real	threads que precisam de tempo de resposta bem-definido	96-127

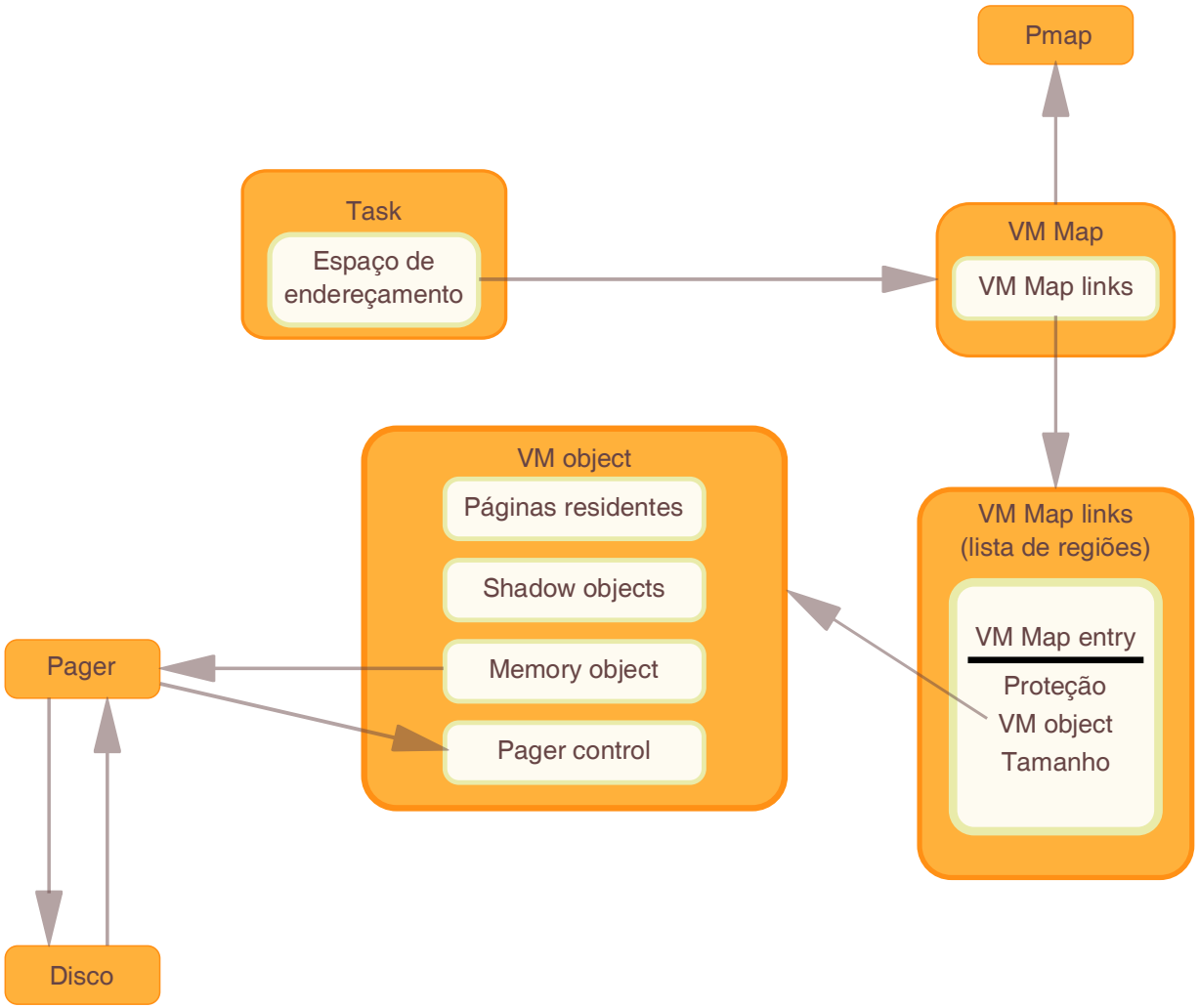
IPC

- Passagem de mensagens
- Canais de comunicação: portas
- Não só entre processos: threads no mesmo processo, threads em processos diferentes e threads no kernel
- Serviços: pipes, signals, exceções, passagem de descritor de arquivo, POSIX IPC
- MIG: traduz syscalls e RPCs em mensagens

Portas

- Base do IPC no Mac OS X
- Cada porta mantém uma fila de mensagens
- Direitos:
  - Cada task tem os *direitos de recebimento* das suas portas
  - Ela pode fornecer *direitos de envio* às outras tasks
- Cada task tem uma porta para se comunicar com o kernel
  - Syscall é "traduzida" (via MIG) em uma mensagem
  - A mensagem é enviada para essa porta

Modelo de memória virtual



Operações

- Page-in
  1. Quando uma *task* é criada, é reservado a ela um espaço de endereçamento (4 GB para 32 bits, 2 PB para 64 bits)
  2. Ao acessar uma nova página, a task gera uma falha de página
  3. O kernel gerencia a falha de página solicitando os dados faltantes ao pager
  4. O pager pega os dados da memória secundária e os fornece ao kernel, que aloca a nova página na memória e atualiza a VM object
- Page-out
  1. O kernel escolhe, usando o algoritmo da segunda chance, a página a ser trocada
  2. Ele envia uma mensagem ao pager, que salva a página na memória secundária

Pagers

- Tipos:
  - `default`: paginação entre memória física e área de swap
  - `device`: paginação para memória de propósito específico (dispositivos)
  - `vnode`: entre memória física e arquivos
    - Roda na camada BSD
- A estrutura do Mach permite que o usuário escreva seus próprios pagers
  - Mas o Mac OS X não fornece essa flexibilidade

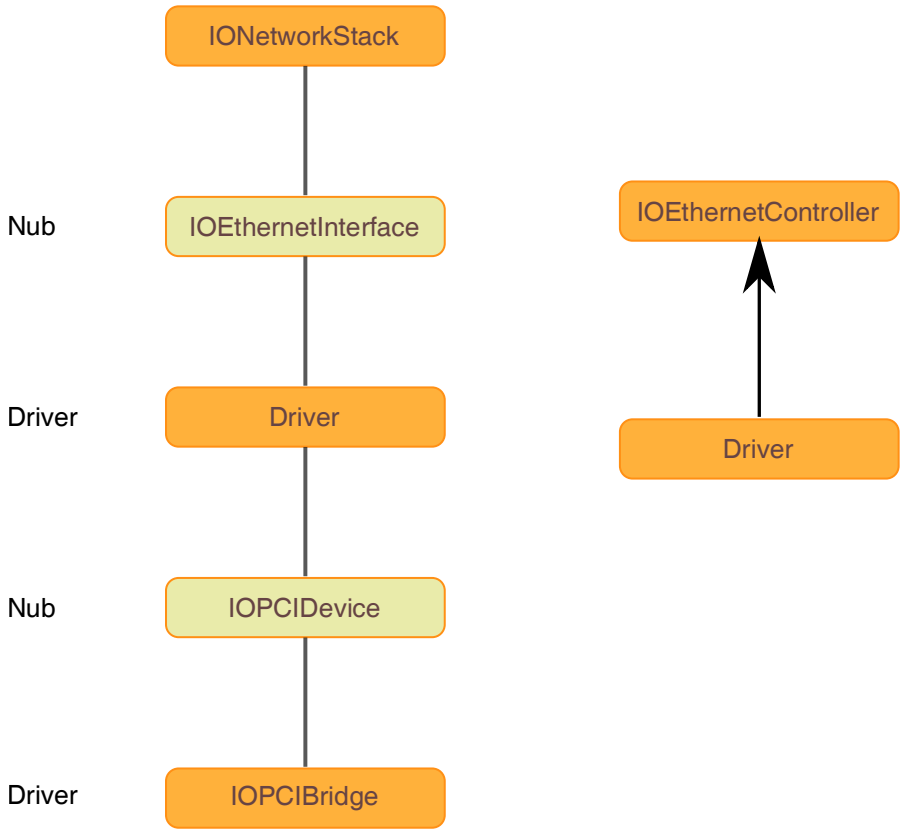
Outros recursos

- O kernel possui o mesmo espaço de endereçamento que aplicações normais
- Copy-on-write (COW): técnica de otimização para redução de overhead na cópia de páginas
  - Fundamental para `fork()`, IPC

I/O Kit

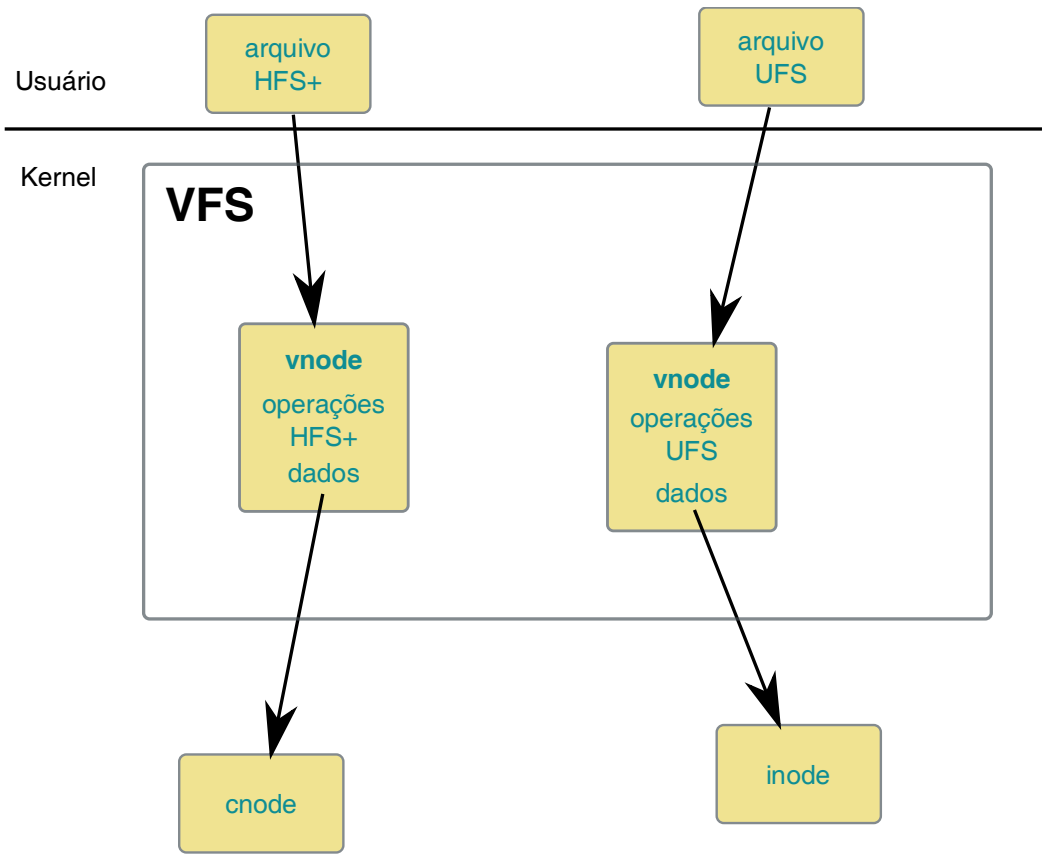
- Modelo simplificado de abstração de hardware simulado pelo paradigma orientado a objetos (C++)
- Cada tipo de dispositivo é abstraído por uma classe em C++ derivada de uma família
- A família não usa detalhes do hardware, mas em geral encapsula informações definidas num padrão
  - Exemplos: porta paralela SCSI, USB
- Drivers se comunicam usando *nubs*
  - Nubs também são usados para encontrar o driver para o dispositivo
- Dispositivos: arquivos em `/dev` (Unix-like)
  - Mas o recomendável é usar a API do I/O Kit

Hierarquia


IONetworkStack
Nub
IOEthernetInterface
Driver
Driver
Nub
IOPCIDevice
Driver
IOPCIBridge
IOEthernetController
Driver

Sistemas de arquivos

- Suporte a diversos sistemas de arquivos: **HFS+**, UFS, NFS, NTFS, ISO 9660, SMB, FAT, UDF
- VFS: provê transparência no acesso aos dados, independente do sistema de arquivos real



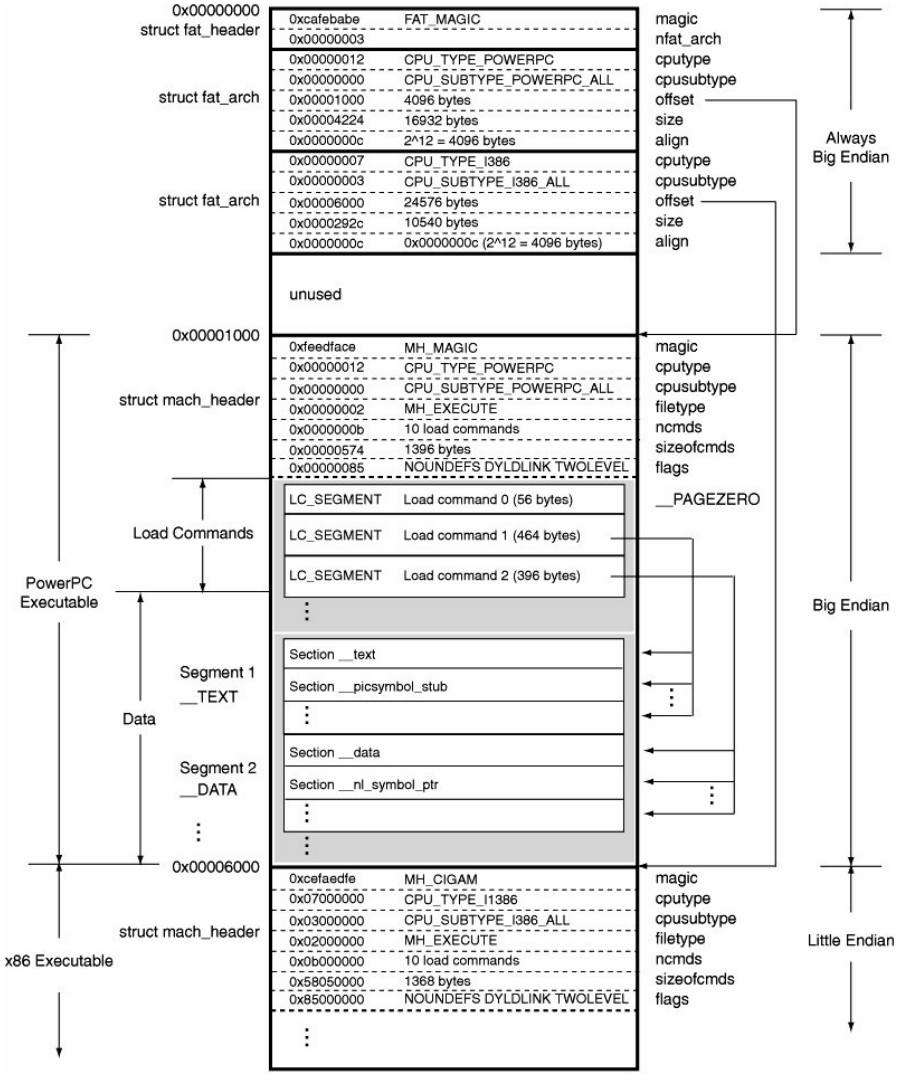
HFS+

- Suporta arquivos de até 2^{63} bytes
- Hierarquia representada por uma árvore B+
  - Extents: regiões do arquivo cujos blocos são alocados contiguamente
  - Os primeiros oito extents do arquivo são guardados nessa árvore
  - Arquivos pequenos com muitos extents são desfragmentados dinamicamente
- Suporta hard links, links simbólicos e aliases
  - Alias: link simbólico que "segue" os movimentos do arquivo
- Permissões de acesso Unix-like e ACLs
- Journaling

Segurança

- Identificadores BSD-like: UIDs e GIDs (proteção de arquivos e processos)
- Direitos de Mach ports (envio/recebimento)
- Memória virtual encriptada (previne análise dinâmica em modo kernel)
- Sistema de auditoria: Log dos eventos de segurança
- ACLs (Access Control List)
- K-auth: avaliação das ACLs pelo Kernel

Binários universais e Rosetta



Referências

- "Inside Mac OS X Kernel", Lucy <whoislucy@gmail.com>
- "Mac OS X Internals: A Systems Approach", Amit Singh
- "Mac OS X Reference Library", http://developer.apple.com/Mac/library/documentation/Darwin/Conceptual/KernelProgramming/About/About.html
- "Modern Operating Systems" (*primeira* edição), Andrew S. Tanenbaum

Perguntas?