

GNU Hurd

MAC0422 — Estudo de Caso

Renato Lui Geh

18 de novembro de 2016

Índice

- 1 História
- 2 Arquitetura Geral
- 3 Microkernel/Mach
- 4 Multiservidor
- 5 Memória
- 6 Escalonamento de Processos
- 7 Sistema de Arquivos
- 8 Comparação
- 9 Referências e Bibliografia

História do GNU/Hurd

1983●	Richard Stallman (RMS) cria o projeto GNU.
1986●	RMS decide usar o TRIX como kernel.
1988●	É decidido usar o Mach como kernel.
1991●	GNU Hurd é anunciado ao público.
1994●	Primeiro boot.
1994●	Emacs e gcc rodam pela primeira vez.
1995●	ext2fs, ftp.
1996●	NFS e GNU Hurd 0.1.
1997●	GNU Hurd 0.2.

História do GNU/Hurd

2011●	GNU Hurd 0.4.
2013●	Debian GNU/Hurd, GNU Hurd 0.5.
2015●	GNU Hurd 0.6.
2016●	GNU Hurd 0.8.

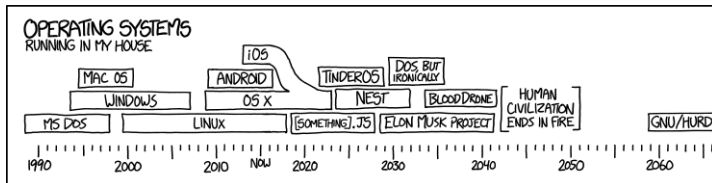


Figura: <https://xkcd.com/1508/>

GNU HURD

HURD: Hird of Unix-Replacing Daemons

HIRD: Hurd of Interfaces Representing Depth

```
GNU HURD := [  
  GNU := GNU's Not Unix  
  HURD := [  
    HIRD := [  
      HURD := [  
        ...  
      ] of Interfaces Representing Depth  
    ] of Unix-Replacing Daemons  
  ]  
]
```

Documentação

- Horrível
- Escassa
- Conversas no IRC
- Confusa
- Alinear
- Feito pela comunidade
- Página de *open issues*
- Recompensas (\$\$\$) para quem resolver

Arquitetura do GNU Hurd

- Microkernel (Mach)
- Multiservidor
- GNU C
- Servidores flexíveis (nível usuário)
- Servidores compilados como quiser
- MIG (Mach Interface Generator)

Kernel

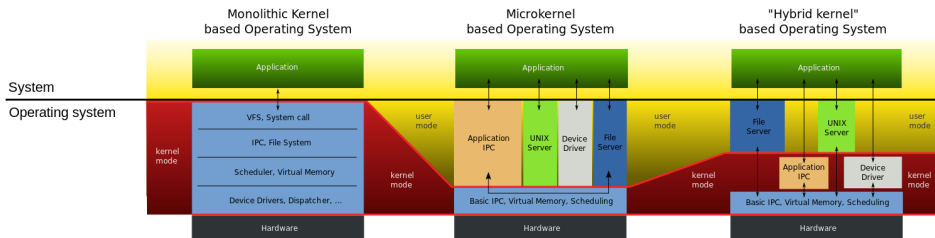


Figura: <https://en.wikipedia.org/wiki/File:OS-structure2.svg>

Comparação

Microkernel (e.g. Mach, kernel do Minix)

- Mais seguro
- Comunicação por mensagens (MIG)
- Responsabilidade bem definida
- Flexível (fácil de debugar os diferentes servidores)
- Mais estável
- Somente o absolutamente necessário

Monolítico (e.g. Linux)

- Não é necessário passar mensagem
- Mais rápido por ter acesso direto
- Código passa a se tornar complicado e confuso (spaghetti code)
- Pequenas (a às vezes aparentemente irrelevantes) mudanças podem quebrar o sistema
- Mais difícil de manter código
- Novos desenvolvedores sofrem tentando aprender código confuso

História do Mach

- Criado em 1984 na Carnegie-Mellon
- Desenvolvimento do Mach terminou em 1994
- GNU Hurd usa GNU Mach (versão modificada para ser *free* e manter-se atualizado)
- Microkernel/Nanokernel (alguns consideram “Hybrid Kernel”)
- Mensagens por meio do MIG (Mach Interface Generator)
- MIG possibilita rodar código por RPC

O que Mach faz e não faz?

Faz:

- Gerenciamento de Memória
- Gerenciamento de Processos
- Comunicações (mensagens dos outros servidores)
- Entrada e Saída

Não faz:

- Sistema de Arquivos
- Drivers
- Aplicativos de Usuário (WM, DE, etc.)

Multiservidor

- Rodam paralelamente ao Mach
- Comunicam-se pelo MIG
- Ficam na camada de usuário
- Qualquer linguagem
- Compilado como quiser
- Debugar enquanto os outros servidores rodam
- Independente de todos os outros servidores e kernel
- Completamente modificável
- Seguro (ficam em camada de usuário)
- Não é preciso dar reboot para testar

Alguns exemplos de servidores

Core

- auth privilégios, senhas e identificação
- crash erros
- exec rodar executáveis
- fifo pipes
- firmlink “half-way between a symbolic link and hard link”
- ifsock sockets
- init boot
- null equivalente a /dev/null e /dev/zero
- procs PIDs
- term terminal
- ...

Filesystem

- ext2fs, isofs, nfs, ufs, ftpfs, storeio

Memória no GNU Mach

- 32-bit
- 2GB kernelspace
- 2GB userspace
- Mach cuida de toda memória virtual
- Pedidos de memória pelos processos feitos por mensagens
- Mach retorna um ponteiro para a memória e o tamanho alocado

Escalonamento

Gerenciamento de memória por meio de uma Red-Black Tree (RBT) e usando Best-fit.

Red-black tree é uma BST com espaço $\mathcal{O}(n)$ e operações $\mathcal{O}(\log n)$.

Blocos: lista duplamente ligada.

Usa-se uma RBT para ordenar os blocos em tamanho. Acha o melhor fit em tempo $\mathcal{O}(\log n)$. Modificar a lista é feito em $\mathcal{O}(1)$ já que é duplamente ligada.

Complexidade final é $\mathcal{O}(\log n)$.

(BSD, Linux, entre outros)

Processos no GNU Mach

Kernelspace:

- Usa *continuations* (estrutura que guarda informação do processo)
- Continuations usam menos espaço pois não precisam da pilha de threads do kernel
- Não é preemptivo (por causa de continuations)
- Ser preemptivo requer toda a pilha de threads do kernel
- Escalonamento feito pelo Mach
- Escalonamento 1:1 (igual a Solaris, NetBSD, FreeBSD, OS X, iOS, OS/2 e Win32)
- Escalonamento feito por filas de prioridade

Processos no GNU Mach

Userspace:

- Threads
- libthreads
- libports
- Servidores tem maior prioridade que user threads
- Não suporta múltiplas CPUs (anos 90)

Filesystem no GNU Hurd

Translators:

- Por *i-nodes*
- Vários tradutores (servidores) para fs
- Cada tradutor é um tipo de formato
- Exemplos: ext2fs, fatfs, ufs, isofs
- Uma busca referencia uma port (uma porta para comunicar-se com o kernel)

Filesystem no GNU Hurd

A ideia: ao invés de uma lista telefônica contendo arquivos, temos uma árvore de busca que retorna uma porta para um servidor que pode acessar o arquivo.

“So we don’t have a single phone book listing all servers, but rather a tree of servers keeping track of each other. That’s really like calling your friend and asking for the phone number of the blond girl at the party yesterday. He might refer you to a friend who hopefully knows more about it. Then you have to retry.”

— Markus Brinkmann

Filesystem no GNU Hurd

Usando-se uma árvore de busca de *translators* possibilita algumas coisa bem interessantes:

- `nfs`
- `ftpfs`

Isso significa acesso transparente a servidores FTP no próprio filesystem. Usando-se o tradutor `ftpfs`, o usuário não precisa se preocupar com protocolos de rede.

Referências e Bibliografia I



Marcus Brinkmann. *The Hurd, a presentation by Markus Brinkmann*. URL:

<https://www.gnu.org/software/hurd/hurd-talk.html>.



GNU Org. <https://www.gnu.org/software/hurd/hurd.html>. URL:

<https://www.gnu.org/software/hurd/hurd.html>.