PCS3446 – Sistemas Operacionais

Prof. João José Neto

Estudos de Casos

Aula 21 – Sistema Mach

Sistema Mach – Histórico

- **Projeto Mach:**
  - Carnegie-Mellon University, 1985-1994 – até o **Mach 3**
  - Utah University – **Mach 4**
- **Origens:** no sistema **RIG (Rochester Intelligent Gateway)**, Universidade de Rochester, **1975**
- A **primeira versão** do sistema Mach foi lançada em **1986** para o computador **VAX 11/784**, multiprocessador com **quatro processadores**

Sistema Mach

Logo em seguida, foi **portado** para **IBM PC/RT** e para **Sun 3, máquinas de porte muito menor.**

Sistema Mach

- Em **1987** estava implantado também em outros multiprocessadores: **Encore** e **Sequent**

Nesta época, o Mach estava **concebido para ser usado em um único computador**

Mach 2.5

- A ***Open Software Foundation*** usou o sistema **Mach 2.5** como base para seu primeiro sistema operacional: **OSF/1**
- Em **1988**, o **Mach 2.5** apresentava um ***kernel* monolítico e grande**.

Mach 3.0

- Em **1989**, a ***Carnegie Mellon University*** reduziu o volume do *kernel*, transferindo boa parte desse software para a área do usuário, e mantendo somente o ***microkernel* do Mach.3.0**
- Na presente apresentação resume-se apenas a **versão 3.0**, base de todas as versões do sistema Mach que foram produzidas pela ***Open Software Foundation***.

Mach 3.0 – Objetivos

- Pesquisar organizações de S.O. que proporcionassem **suporte eficiente para a passagem de mensagens** entre os processos.
- Suportar **comunidades de processos** em ambientes **multiprocessados,** cuja **única forma de comunicação** possível seja a **mensagem**.
- Servir de **base** para a construção de **outros sistemas operacionais**.

Mach 3.0 – Objetivos

- Dar suporte a **espaços de endereçamento grandes e esparsos**.
- Proporcionar **acesso transparente a recursos de rede**.
- Explorar o **paralelismo** do sistema e dos aplicativos.
- Tornar o sistema Mach o mais **portátil** possível fazendo-o compatível com um grande número de máquinas.

Mach 3.0 – Características

- **Pesquisa e desenvolvimento**:
  - sistemas **distribuídos**
  - sistemas **multiprocessados**
  - **emulação de outros sistemas operacionais**, tais como UNIX, MS-DOS, Macintosh Operating System.
- Até os dias de hoje, o **Mach** continua sendo um dos poucos sistemas multiprocessados **totalmente independentes do hardware** em que são executados.

Microkernel

- Este núcleo do Sistema Mach foi projetado de forma tal que possa **emular vários Sistemas Operacionais**, tais como o UNIX, o MS-DOS, e outros.
- **Administra**: serviços de Entrada e Saída, processos, comunicações, memória.
- **Arquivos**: os sistemas de arquivos, diretórios, etc. são tratados fora do núcleo, na **área do usuário**.
- Provê todos os recursos necessários à operação do sistema, mas não implementa as suas **políticas,** deixando-as também para serem tratadas fora do núcleo, **no nível do usuário.**

Abstrações

- Oferece **cinco abstrações** básicas:
  - **Processo** (conjunto de threads relacionadas entre si)
  - ***Thread*** (unidade de scheduling)
  - **Objeto** de memória (unidade de mapeamento)
  - ***Port*** (dispositivo de conexão entre elementos do sistema)
  - **Mensagem** (única forma de comunicação no sistema)
- **Outras abstrações** são também gerenciadas pelo sistema, tanto as relacionadas à operação básica, como também outras, externas ao modelo.

Processos e *Threads*

- **Processos**: são ambientes básicos nos quais ocorre a execução dos programas. Os processos constituem as unidades básicas de alocação de recursos no Sistema Operacional Mach.
- ***Threads:*** no Mach, *threads* são **entidades executáveis**, que formam **parte de um único processo** (os processos Mach que forem limitados a um *thread* apenas correspondem aos processos clássicos estudados na teoria).

Objetos de memória

- **Objetos de memória:** São elementos característicos do **S.O. Mach,** e consistem de estruturas de dados que podem ser **mapeadas** para alguma parte do espaço de endereçamento dos processos.
- Objetos de memória do Mach podem ocupar **uma ou mais páginas** da área de endereçamento.
- Os objetos de memória constituem a **base do sistema de memória virtual** no sistema Mach.

Interrupções da memória virtual

- As **interrupções** ocorrem quando é feita uma referência a algum objeto de memória não presente na memória física.
- O *microkernel* apenas **captura** a interrupção.
- O tratamento dessa interrupção fica a cargo de um programa na área do usuário, para o qual **a ocorrência é encaminhada através de uma mensagem** enviada pelo sistema.

Comunicação

Toda a comunicação do Mach é feita segundo o seguinte mecanismo:

- É baseada exclusivamente na **passagem de mensagens** entre processos, efetuada com o auxílio de *ports*.
- ***Port*** é uma caixa de correio protegida, situada no *microkernel*, que guarda uma **lista ordenada das mensagens** endereçadas ao processo ao qual pertence o *port*.

Comunicação

- As filas não têm tamanho fixo, mas **bloqueiam o remetente** caso atinjam um determinado comprimento (que pode ser **escolhido** convenientemente, caso a caso).
- Direitos de acesso (***capabilities***) ao *port* podem ser repassados a outros processos.
- Os processos com direito de acesso enviam **mensagens** ao *port* adequado, e este pode então ser lido pelo respectivo proprietário.

MACH BSD UNIX SERVER

Mach BSD UNIX Server

(BSD = Berkeley Software Distribution)

- O *microkernel* permite que sejam **emulados** outros sistemas operacionais, que são executados sem privilégios, como programas do **usuário**.
- O núcleo do sistema operacional emulado compõe-se de duas partes independentes:
  - Uma, responsável pelo **manuseio de recursos** do sistema
  - Outra, responsável por tratar as **chamadas de supervisor** (inspirado no esquema primeiramente empregado pelo IBM OS-VM/370, em que vários sistemas operacionais podiam conviver).
- Esse esquema **separa as funções** e torna os Sistemas Operacionais mais simples e fáceis de manter.
- Proporciona também **independência de plataforma** para os Sistemas Operacionais, confinando as dependências no *microkernel* do Mach.

- Pode-se assim, **na mesma máquina, ao mesmo tempo, executar programas em dois ou mais Sistemas Operacionais**, permitindo, por exemplo, testar um novo sistema operacional ao mesmo tempo que outro, já testado, opera normalmente.
- Ganha-se também, desta maneira, a possibilidade de **executar programas de tempo real** (esta é uma importante diferença, pois o UNIX original não fornecia recursos suficientes para dar suporte a sistemas de tempo real).
- Aumenta a **segurança** ao proporcionar o recurso de **intercâmbio** de arquivos entre programas executados em **diferentes versões** dos sistemas operacionais considerados.

Gerenciamento de Processos no Mach

- **Processos**
  (espaço de endereçamento + coleção de *threads*)
  são **passivos**, e coletam recursos para permitir a execução de *threads* a eles associados.
- ***Thread*** é um conceito estritamente relacionado com a **execução**.
- ***Ports***: há vários tipos de ports no Mach:
  - ***process*** *port* (para **comunicação** com *kernel*),
  - ***bootstrap*** *port* (específico **para dar partida** aos processos),
  - ***exception*** *port* (destinado a reportar **erros**),
  - ***registered*** *ports* (ocupam-se da comunicação com **servidores**).
- **Primitivas**: operações básicas usuais de um sistema operacional:
  - *create, terminate,*
  - *suspend, resume,*
  - *priority, assign,*
  - *info, threads.*

Threads

- São os elementos **ativos** do sistema operacional Mach, aos quais o *kernel* aplica algumas operações:
  - **criar, eliminar,**
  - **multiprogramar,**
  - **bloquear,** etc.
- **Compartilham** entre si os recursos **do processo a eles associado**
- Podem, além disso, possuir **recursos próprios**, exclusivamente associados a eles

- Um **semáforo** *contador* indica se o *thread* está executável (contador = 0) ou não (contador > 0)
- Disponibiliza algumas **chamadas de supervisor,** para controle direto dos threads: *fork, exit, join, detach* (nunca ninguém irá ficar esperando por este *thread*), *yield* (libera espontaneamente o processador), *self* (informa ao thread solicitante sua identificação)
- **Sincronização**: usa *mutexes* e variáveis de decisão: *lock, trylock, unlock*

Scheduling

- Influenciado pelo **multiprocessamento**.
- Quando um *thread* se bloqueia, termina ou esgota seu *quantum*, há um **chaveamento de *thread*** na **fila local** do processador.
- Se a **fila global** estiver vazia, seu acesso é bloqueado, e só então é feito o chaveamento.
- Em todos os casos, a **prioridade** dos *threads* tem influência na sua ordem de execução.
- ***Threads* cooperativos**: por suas particularidades, a sua **sincronização** é controlada pelo usuário, que para isso emprega **prioridades** e **mensagens**.

Gerenciamento de memória

- O gerenciamento de memória no Sistema Mach é baseado no princípio da **paginação**.
- **Separa** as partes do gerenciamento de memória **dependente** e **independente do hardware**.
- Isso torna o gerenciamento de memória mais **portátil**.
- Interage fortemente com o mecanismo de **comunicação**.

Código

- O **Código** dos programas no Mach se divide em **três partes**:
  - **pmap** (referente ao **MMU**, portanto, no *kernel*),
  - parte referente ao gerenciamento de **falta de páginas** (interrupções)
  - o ***memory manager*** ou ***external pager***, referente ao gerenciamento da **memória virtual** em disco.
- O uso de um **protocolo bem definido** com o *kernel* facilita a criação de novos gerenciadores.

Memória virtual

- **Memória virtual**: as instruções de referência à memória utilizam um campo de endereçamento de 32 bits, de endereço paginado, esparso, dividido em regiões alocadas/não alocadas, permitindo dessa forma o surgimento de regiões vazias entre elas.

- ***Memory object*** cria um mapa de memória, aplicável a uma região não ocupada, e pode ser ocupado tanto por uma região da **memória principal** de um programa como por parte de um **arquivo**.
- Se o *memory object* for usado para conter parte de um **arquivo**, em lugar de ser acessado através de operações de **entrada / saída em disco**, este poderá ser diretamente lido ou gravado mediante o **uso de instruções de acesso à memória** (ou seja, de forma muito mais eficiente).

- **Primitivas**:
  - *allocate, deallocate,*
  - *map, copy, inherit,*
  - *read, write*.
- **Compartilhamento de memória**
  - no mesmo processador, e
  - entre processadores.

Esta apresentação foi baseada em

Tanenbaum, *Distributed Operating Systems,* cap.8

Prentice Hall, 1995