Universidade de São Paulo
Instituto de Matemática e Estatística

MAC5755
Sistemas Operacionais Distribuídos

Sistema Operacional Inferno

Cleber Miranda Barboza - N^{o}USP:3286353
e-mail: `cleberc at linux.ime.usp.br`

21 de novembro de 2003

1 Introdução

O objetivo desse texto é apresentar uma visão geral do que é o Sistema Operacional *Inferno* e alguns requisitos para a sua utilização.

Assim, o *Inferno* é um sistema operacional, desenvolvido pela Lucent Technologies' Bell Labs, que visa fornecer facilidades para o desenvolvimento e a execução de:

- **serviços distribuídos**;
- **aplicações de rede**.

Os recursos necessários para a sua utilização irão depender do tipo de aplicação a ser utilizada:

- **1MB -** Para sistemas simples que possuam apenas um dispositivo como um leitor ótico, por exemplo;
- **4MB -** Para sistemas com mais de um dispositivo, mas que não haja a necessidade de ambientes gráficos;
- **16MB no mínimo -** Para sistemas mais complexos em que há vários dispositivos e faz-se o uso de ambientes gráficos.

2 Principais Características

As principais características do *Inferno* são:

- **portabilidade através de processadores -** Utilização do sistema nas arquiteturas:
  - Intel;
  - SPARC;

  - MIPS;
  - PowerPC.

- **portabilidade através de ambientes -** Utilizado das seguintes maneiras:
  - sistema Operacional Nativo;
  - hospedado dentro dos seguintes sistemas:
    * Windows;
    * Unix (Irix, Solaris, FreeBSD, Linux, AIX, HP/UX);
    * Plan 9.

- **adaptabilidade dinâmica -** Dependendo do hardware e dos recursos diponíveis, diferentes módulos poderiam ser carregados para a realização de funcionalidades específicas. Um video-player poderia carregar diferentes decodificadores, por exemplo;

- **aplicações portáveis -** Através da linguagem de programação *Limbo*, fornecidade pelo *Inferno*, é possível desenvolver aplicações que possam ser utilizadas através das diferentes plataformas.

3 Visão Geral

O inferno teve grande influência do sistema operacional **Plan 9**, que é baseado em três princípios:

- **recursos como arquivos**;
- **espaço de nomes**;
- **protocolo de comunicação padrão**.

3.1 Recursos como arquivos

Todo recurso disponível no sistema é visto como arquivo, não importando se é local ou remoto. Até mesmo "processos", conceito presente na maior parte dos sistemas operacionais, são representados como recursos e, portanto, também são vistos como arquivos. O que o *Inferno* faz na verdade é levar o conceito de recursos como arquivos ao extremo. O acesso a esses recursos é feito através das operações: *open, close, read, write.*

As principais vantagens relacionadas a essa abordagem são:

- **interface simples e bem definida -** Através de operações simples é possível acessar os recursos e suas funcionalidades uniformemente;

- **alta portabilidade -** Como todo recurso é visto como um arquivo, a facilidade na portabilidade torna-se evidente;

- **segurança -** Essa visão sobre os recursos permite que os aspectos relacionados a segurança sejam focados no tratamento de arquivos.

A seguir, temos a localização de onde se encontram os principais recursos no sistema *Inferno*:

- interface de rede: */dev/tcp*, */dev/udp*, etc;
- informações de processos: */prog*;
- sistema de janelas: */dev/draw*;
- informações: */dev/user*, */dev/time*, */dev/sysname*, */dev/random*.

É possível associar um dado recurso a mais de um arquivo. Quando isso é feito, cria-se um diretório, colocando os arquivos relacionados ao recurso dentro desse diretório. Tipicamente, cada diretório poderia ter dois arquivos: um para o controle de estados do recurso e outro para operações de entrada e saída relacionados ao recurso. Por exemplo, o diretório */dev/tcp* representa o recurso relacionado ao protocolo TCP/IP. Nesse diretório, podemos encontrar dois arquivos:

- **data -** Utilizado para operações de entrada e saída;
- **ctl -** Utilizado para o controle de estado desse recurso.

3.2 Espaço de nomes

Espaço de nomes serve para manter a representação uniforme de recursos.

Assim, o que se faz é agrupar recursos, formando uma representação hierárquica semelhante ao sistema de arquivos hierárquico presente em sistemas *UNIX*.

Espaço de nomes pode ser:

- **importado**;
- **exportado**.

Vale citar que a localização do espaço de nomes é transparente, não importando se é local ou remoto; isso significa que as aplicações vêem o espaço de nomes da mesma maneira. Essa transparência é feita com o uso de um protocolo chamado *Styx* que será detalhado em 3.3.

3.3 Protocolo de comunicação

Styx é protocolo para apresentação de recursos utilizado pelo *Inferno*. Ele é uma variação do protocolo 9P desenvolvido para o Plan 9 o qual possui a seguinte idédia básica:

- *Codificar operações de arquivos em mensagens para serem transmitidas via rede.*

O protocolo garante transparência completa de recursos, ou seja, usuários (desenvolvedores de aplicações) não vêem o protocolo, mas apenas arquivos.

Ele é acima e independente da camada de comunicação (TCP/IP, ATM, PPP, infravermelho, etc)

Além disso, é o Styx quem provê:

- visão hierárquica de recursos;
- informações de acesso: permissões, tamanhos e datas de arquivos (recursos);
- semântica para leitura e escrita.

3.4 Limbo

Limbo é a linguagem de programação para o Inferno. Dentre suas caraterísticas, temos que sua sintaxe foi influenciada pelo C e Pascal, além de apresentar um compilador semelhante ao do Java, pois o código objeto gerado (bytecode - aquivo .dis) é independente de máquina; a interpretação do código é feita por uma Máquina Virtual (Dis) - segredo da portabilidade das aplicações.

A programação utilizando-se o Limbo é feita através do conceito de módulos, ou seja, um programa Limbo é composto por um conjunto de módulos que cooperam para realizar uma tarefa. Um módulo consiste basicamente de duas partes:

- especificação das interfaces públicas (funções, constantes, tipos abstratos de dados, etc);
- código que implementa as interfaces.

Os módulos são carregados dinamicamente utilizando-se o operador *load*.

Outra característica apresentada pela linguagem é a checagem de tipagem rígida em tempo de execução e compilação; a checagem de tipos em tempo de execução é feita sobre os modulos no instante em que são carregados.

3.5 Dis

Dis é a Máquina Virtual (MV) do Inferno. Além de interpretar o código objeto gerado pelo compilador do Limbo, foi desenvolvido também para compilação on-the-fly (just-in-time), ou seja, compilação em tempo de execução.

O design da MV envolve:

- conjunto de instruções;
- sistema de módulos.

As instruções do código objeto seguem o modelo CISC (Complex Instruction Set Computer), apresentando o seguinte formato:

OP src1, src2, dst

Como exemplo, temos que a operação $c = a + b$, utilizando a forma acima, ficaria assim:

add a, b, c

Ale de *add*, existem instruções para alocar memória, carregar módulos, criar processos, sincronização e comunicação entre processos.

3.5.1 Gerenciamento de memória

O gerenciamento de memória está ligado ao conjunto de instruções da MV. Ele é feito através do uso de um coletor de lixo híbrido:

- contagem de referências;

- real-time sweeping (mark-and-sweep - "marcar e limpar"), três passos:
  - para todo objeto no sistema, se ele tem uma marca, ela é limpa;
  - através das pilhas de execução, encontra-se os objetos que estão sendo referenciados, marcando-os;
  - no passo final, percorre-se o heap linearmente, removendo todos os objetos que não possuem a marca;

3.6 Segurança

O Inferno provê segurança de:

- comunicação;
- controle de recursos;
- integridade de sistema.

A segurança na comunicação é feita através da existência do conceito de canais de comunicação entre processos e *Threads* (processos leves).

Um canal de comunicação nada mais é do que um tipo de dado presente na linguagem Limbo que pode ser utilizado pelo programador.

O sistema *Inferno* pode criptografar e decriptografar as mensagens enviadas por um canal, tornando esse mecanismo transparente para as aplicações.

Há também a presença de mecanismos de autenticação para a inclusão e exclusão de recursos ao sistema. Além disso, os recursos são acessados somente por chamadas de módulos que os provê.

Alguns algoritmos de criptografia presentes no sistema *Inferno* são: SHA, MD4, MD5, Elgamal (assinaturas), RC4, DES, Diffie-Hellman (chave pública).

3.7 Inferno/Limbo vs. JavaOS/Java

Dentre as características que se assemelham e se diferenciam, temos:

- ambos possuem sintaxe influenciada pelo C;
- utilização de uma máquina virtual por ambos;
- Java usa o conceito de objetos;
- Limbo é um pouco mais simples, entretanto provê alguns tipos de dados sofisticados, como o *channel* (utilizado para comunicação entre processos e *Threads*), além de mecanismos para controle de concorrência, autenticação, segurança, etc;
- biblioteca gráfica do Inferno (Tk) mais completa que o AWT.
- Máquina virtual
  - A arquitetura do inferno segue o modelo de transferência de memória (*memory-to-memory*), ou seja, as instruções do Dis são traduzidas para uma única instrução de máquina (CISC).

– Já a JVM (Java Vitual Machine) usa a arquitetura de pilha.
Utizando o exemplo c = a + b, teriamos o seguinte conjunto de instruções:
push a
push b
add
store c

3.8 Inferno - Hoje

Hoje o inferno se encontra em sua 4.a edição, apresentando desenvolvimento em passos lentos. Entretanto, há planos para realizar integração com o Java

Referências

[1] Vitua Nova.*Inferno Overview*, em: http://www.vitanuova.com/inferno/papers/bltj.html

[2] Sean Dorward, Rob Pike, David Leo Presotto, Dennis M. Ritchie, Howard Trickey, Phil Winterbottom. *The Inferno Operating System*, em: http://www.vitanuova.com/inferno/papers/bltj.html

[3] Rob Pike, Dennis M. Ritchie. *The Styx Architecture for Distributed Systems*, em: http://www.vitanuova.com/inferno/papers/styx.html

[4] Dennis M. Ritchie. *The Limbo Programming Language*, em: http://www.vitanuova.com/inferno/papers/limbo.html

[5] Phil Winterbottom Rob Pike Bell Labs. *The design of the Inferno virtual machine*, em: http://www.vitanuova.com/inferno/papers/hotchips.html