Sistema Operacional Unix

André F. Silveira, Gustavo H. P. Kwaczynski,Lucas E. De Matte, Rodrigo Miss

Universidade Tecnológica Federal do Paraná (UTFPR)

Guarapuava – Pr - Brasil

Abstract. *Unix is an operating system with multitasking and multiuser platform, its core was written in "C" language in the remaining assembly, multithreaded applications was implemented only in the newer versions, has shared memory and semaphores. There are two levels of access (user and kernel), your file structure is hierarchical, it has easy access to input and output peripherals, allows interruptions by the devices, your system process scheduling happens through the Round Robin system with Priorities. The manages memory in the latest version adopts demand paging*

Key-words: *Kernel, Manages, language, system, operational.*

Resumo. *Unix é um sistema Operacional com plataforma Multiusuário e Multitarefa, seu núcleo foi escrito em linguagem "C" o restante em assembly, aplicações de multithreads somente foi implantada nas versões mais recentes, possui memória compartilhada e semáforos. Existem dois níveis de acesso (usuário e kernel), sua estrutura de arquivos é hierárquica, tem um acesso simples com periféricos de entrada e saída, permite interrupções por parte dos dispositivos, seu sistema de escalonamento de processos acontece por meio do Round Robin com sistema de Prioridades. A gerencia de memoria na versão mais recente adota a paginação por demanda.*

Palavras-chave: *Kernel, Gerencia, linguagem, sistema, operacional.*

1.Introdução

Neste artigo será abordado a historia, a estrutura e a organização do sistema operacional UNIX bem como um resumo de seu nascimento, empresas envolvidas sua disseminação e contexto no cenário mundial.

O UNIX Foi o precursor de diversos sistemas operacionais tais como o Linux, Minix, Solaris, Mac OS X, e diversos outros pequenos sistemas como por exemplo sistemas automotivos, e pequenas maquinas, isto devido a sua facilidade de implementação, código aberto, bibliotecas o que permite uma maior facilidade em reescreve-lo ou molda-lo,

O sistema UNIX foi desenvolvido com o objetivo inicial de proporcional um ambiente onde os programadores pudessem criar novos programas, mas devido a sua grande popularidade logo já estava em uso por universidades e

empresas, e com a rápida disseminação e logo havia varias implementações diferentes ou seja varias adaptações do sistema.

Pode-se dizer que cada usuário UNIX tem um sistema diferente isso ocorre por que o UNIX é separado por módulos, ou seja se um usuário não usa todos os módulos do sistema ele pode simplesmente remove-lo sem causar danos ao funcionamento do sistema, pode também acrescentar novos módulos, criar, ou expandi-los deixando da melhor maneira possível para seu trabalho.

O Sistema UNIX é:

- **Multitarefa** - É possível, executar vários programas, controlar dispositivos de entrada e saída, gerenciar a performance do sistema, compilar novos programas e editar arquivos ou diretórios.
- **Multiusuário** - É possível, que vários usuários acessem o sistema simultaneamente, ou seja um sistema pode estar conectado a outras maquinas e estas podem abrir o terminal e cada usuário mandar um comando para o sistema.
- **Sistema de Shell** – Shell é um ambiente que interpreta e executa comandos, ele pode receber um só comando e executa-lo, como pode receber vários comandos de diversos usuários e criar um pipe e executa-los um a um.
- **Sistema de arquivos** – Provavelmente o mais importante componente de um sistema, o formato de organização dos arquivos, no UNIX o sistema de arquivos é hierárquico de diretórios ou pastas, seu formato final da-se como um organograma.

2.História do Unix

Em 1960 as organizações MIT[1], Bell Labs[2] e General Eletric se uniram para a criação de um novo sistema operacional que viesse a substituir os sistemas Batch, com o andar do projeto a empresa Bell Labs se retirou mas dois de seus pesquisadores Ken Thomposon e Dennis Ritchie continuam o projeto e vieram a desenvolver o sistema UNICS que no futuro se tornaria o sistema UNIX.

Após o Sistema já estar operacional a empresa Bell Labs que era uma subsidiaria da AT&T não podia comercializar o mais novo sistema devido as leis de Antimonopolio. Em 1974 os criadores do UNIX lançaram um documento descrevendo o novo sistema, o que despertou a curiosidade das Universidades em fazer uso deste moderno Sistema operacional, oque foi útil a AT&T já que as universidades poderiam licenciar o UNIX, foi ai que a Universidade de Berkeley na Califórnia recebeu copias do sistema bem como seu código fonte, e realizaram diversas modificações criando novas releases. Grandes melhorias correram com as modificações da universidade entre elas a criação do Protocolo TCP/IP, memoria virtual , Shell entre outras melhorias.

_____

[1] MIT – Massachusetts Institute of Technology.

[2] Bell Labs – Bell Telephone Laboratories.

Com um sistema mais instável e melhorias constantes as empresas abriram as suas portas e inseriram o UNIX em seus produtos entre as grandes estão HP e IBM entre outras. Em 1982 a AT&T é liberada para comercializar o sistema, a partir dai a AT&T começou a lançar novas atualizações bem como a Universidade de Berkeley, e nunca
houve uma unificação entre os dois apesar de usarem a mesma base. A AT&T continuou sua comercialização com o sucesso da release (SVR4), já Berkeley criou a Open Software Foundation, disponibilizando o seu sistema de forma livre.

3.Características

3.1. Sistema operacional multitarefas

Sistemas operacionais multitarefas são sistemas capazes de executar um ou mais processos simultaneamente, enquanto sistemas monotarefas permitem executar apenas um processo por vez, sendo executado sequencialmente de forma tão rápida causando a falsa ilusão de que estão sendo executados simultaneamente.

O unix é um sistema operacional multitarefa com preempção, significando dizer que um intervalo de tempo chamado **quantum** é que define o tempo que cada processo possui para utilizar o processador para executar suas tarefas. Após esgotar o quantum, o unix suspende a execução do processo e salva as informações necessárias para posterior execução (contexto do processo), e coloca em execução o próximo processo da fila de espera.

3.2. Sistema operacional multiusuários

O unix é um sistema operacional multiusuário, pois permite que dois ou mais usuários utilizem o mesmo computador simultaneamente para executar diferentes aplicações concorrentemente e independentemente, geralmente através de terminais conectados ao determinado computador.

O unix possui dois tipos de usuários, o usuário root ou superusuário que possui total permissão sobre o sistema, e os usuários comuns que possuem permissões limitados e configuráveis.

4. Estrutura do Sistema

Quase em totalidade o código fonte do sistema Unix foi desenvolvido em linguagem C, sendo a minoria restante em assembly, favorecendo o sistema quando se deseja fazer a portabilidade para outras plataformas de hardwares.

A estrutura do Unix é baseada em camadas, e o modo de acesso ao sistema se dá por meio de dois nível, usuário e kernel.

4.1.Kernel

O kernel é a uma fatia do sistema que é responsável por controlar o hardware e fornecer as system calls para que as aplicações do usuário tenham acesso às funções do sistema.

O kernel possui uma parte dependente do hardware ou outra independente. A parte dependente do hardware está atrelada as rotinas de tratamento de interrupções e exceções, drivers e tratamento de sinais, portanto, deve ser reescrita quando estiver sendo portado o sistema Unix para outra plataforma de hardware. A parte independente de hardware não possui nenhum vínculo com a plataforma que está portada, e é responsável pelo tratamento das system calls, gerência de processos, gerência de memória, escalonamento, pipes, paginação, swapping e sistemas de arquivos.

O kernel do Unix, se comparado com os demais sistemas operacionais, é dotado de um conjunto pequeno de system calls, porém, a partir delas podem ser desenvolvidas funções de enorme complexidade, sendo uma estratégia ao ser criado o Unix, pois novos utilitários podem ser facilmente integrados ao sistema, sem que o kernel sofra qualquer alterações.

4.2.Biblioteca Padrão

Para cada rotina do sistema, existe uma função na biblioteca padrão do Unix, que permite ocultar os detalhes do modo de acesso usuário/kernel ao sistema. A biblioteca fornece uma interface entre os programas e o sistema operacional, fazendo com que as chamadas de sistemas sejam invocadas.

4.3.Utilitários

A camada de mais alto nível é a interface com o usuário, é nela que está implementado vários softwares, como editores de texto, navegadores de internet, compiladores, e o shell.

O shell é um interpretador de comandos, responsável por ler e verificar a sintaxe dos comandos introduzidos pelo usuário e finalmente passar o controle para outros programas que realizaram a função solicitada. O Unix possui três interpretadores de comandos populares, a saber, Bourne Shell (sh) que foi o primeiro shell disponível e está presente em todas as versões do Unix, o C Shell (csh) e o Korn Shell.

Além das várias interfaces em modo texto, também existem várias interfaces gráficas, como por exemplo a X Windows System.

5. Sistema de Arquivos

A forma de armazenamento dos arquivos no UNIX é única pois não há outro sistema operacional que mantenha uma estrutura parecida. Os Arquivos são centrais, comandos e até a comunicação com o processador ocorrem com entidades semelhantes a arquivos, os arquivos podem pertencer a uma ou mais categorias sendo elas “*Group”*- pertence a um grupo; “*User”*- Pertence a um usuário; “*Others”*- Não pertence a nem um grupo e a nenhum usuário.

A organização dos diretórios é chamada de *filesystem,* sua estrutura lembra uma arvore, um organograma, chamado de *root directory e se inicia pela “/”,* programas executáveis são alocados na pasta “/bin” arquivos provenientes de dispositivos de entrada e saída “/dev”, bibliotecas “/lib” e usuários “/usr”. Como no UNIX praticamente tudo é arquivo é necessário um sistema de propriedade e proteção, que garantem a integridade dos arquivos. Para o site ma cada arquivo é somente um conjunto de bytes, pois ele não diferencia um arquivo de texto de um executável, a função do sistema é

simplesmente gerar o acesso a estes arquivos, a a interpretação dos mesmos fica por parte da aplicação.

A localização de um arquivo no diretório é indicada através de um *pathname* que pode ser relativo ou absoluto, *pathname* relativo é a partir de onde o usurário se encontra e absoluto é a localização completa partindo do "/".Uma curiosidade do sistema de arquivos do UNIX é que ele mantem a sua estrutura logica mesmo sendo adicionados mais discos físicos, ou seja independente de quantos discos o computador tenha, para o usuário é como se existisse somente um.

As informações dos arquivos ficam armazenados nos i-nodes, existe um bloco onde os i-nodes ficam, eles armazenam a data de criação, modificação, inserção entre outras informações do arquivo, cada i-node possui 64 bytes para armazenar estas informações, quando um arquivo possui uma enorme quantidade de informações, extrapolando o tamanho dos i-nodes este é dividido entre quantos forem necessários, e ao fim de cada i-node existe uma especie de ponteiro para o próximo.

6. Gerências

6.1.Gerência do Processador

O sistema UNIX utiliza dois tipos de escalonamento: circular com prioridades e prioridades.

No UNIX, os processos podem ter prioridades entre 0 e 127 (quanto menor o valor, maior a prioridade). Processos do modo usuário geralmente tem prioridades entre 50 e 127, e os do modo kernel entre 0 e 49. Quando estão no estado de pronto, os processos aguardam em filas para serem escalonados, onde cada fila está associada a uma propriedade.

Após escalonado, o processo pode ficar no processador por até 100 milissegundos. Quando o seu quantum termina, o processo retorna para a fila associada à sua prioridade. A prioridade dos processos no estado de pronto é recalculada periodicamente.

6.2.Gerência de Memória

Dentre todos os processos que envolvem o desenvolvimento de um sistema operacional pode-se dizer que a gerencia de memoria é o mais importante e também o mais critico de todos, devido a complexidade dos algorítimos de escalonamento, o alto custo de memorias físicas a necessidade do usuário em ter vários processos em excussão ao mesmo tempo, por isso a maior preocupação que envolve o desenvolvimento da gerencia de memoria é produzir um sistema ágil e que não ocupe muito espaço em memoria.

As primeiras versões do UNIX utilizavam um gerenciamento de memoria do tipo swaping, após a versão 3BSD o sistema adotado para gerenciar a memoria foi paginação por demanda e atualmente é utilizado paginação com swapping.

O espaço de endereçamento da memoria do UNIX é dividido em três partes:

- **Texto:** Parte onde ficam armazenados o código executável dos programas, é uma área protegida contra gravação, ele é estático e pode ser compartilhado utilizando um esquema de memoria compartilhada.

- **Dados:** Diferente da parte de Texto a área de Dados é o local onde se encontram as variáveis de um programa, vetores, é uma área dinâmica e permite alterações e gravações constantes.

- **Pilha:** “Armazena informações do controle de ambiente dos processos e procedimentos, ela cresce dinamicamente do endereço virtual mais alto para o mais baixo”.

A paginação por demanda no UNIX ocorre somente a paginas que são referenciadas em suas execuções, o sistema gerencia duas listas uma contendo as paginas livres, ou seja, que estão disponíveis e outra que contem as paginas já em uso, quando uma pagina é referenciada mas não se encontra na memoria principal ocorre o page fault, esta verificação ocorre através do bit de validade, quando isto ocorre a gerencia de memoria retira uma pagina que esta na lista de livres e carrega ela na memoria e ela passa a constar na lista de paginas em execução.

O Kernel é responsável por implementar a paginação e um segundo processo chamado Daemon é responsável por verificar a lista de paginas livres a cada determinado período de tempo, caso esta lista esteja quase vazia ele é responsável por realizar uma busca nas paginas que estão carregadas e verificar quais estão disponíveis a voltar para a lista de livres.

As paginas de texto podem ser transferidas sem qualquer verificação pois são automaticamente recarregadas quando o arquivo executável é ativado, já as paginas de dados devem ser verificadas se ocorreram modificações, através do bit de modificação, caso o bit informe que ocorreram modificações estas devem ser salvas em disco antes de serem transportadas para a lista de paginas livres.

A substituição de paginas ocorre por um algorítimo FIFO Circular, mas com dois ponteiros em vez de um, como é o comum. O primeiro ponteiro que sempre fica a frente do segundo, e é responsável por desligar o bit de referencias das paginas, já o segundo ponteiro que fica a uma certa distancia do primeiro, é responsável por verificar o estado da pagina, caso a pagina não tenha sido referenciada desde que o primeiro ponteiro passou por ela ela é selecionada para passar a lista de paginas livres.

Se por acaso o sistema não consiga manter um numero minimo de paginas livres a área de swapping é ativada, e ponteiros semelhantes aos anteriores verificam os processos que estão a mais tempo inativo e os seleciona e também faz uma lista dos 4 processos que mais fazem uso da memoria e o que permanecer por mais tempo inativo também é selecionado. Isto ocorre até que a lista de paginas livres volte ao seu tamanho normal.

Para controlar isto uma estrutura com todas estas informações fica fora da memoria junto ao Kernel do sistema verificando o numero de paginas livres e em uso, e realizando a gerencia de memoria.

6.3 Gerencia de Entrada / Saída

Como foi apresentado em Sistema de Arquivos, Cada dispositivo de entrada e saída possui um ou mais aquivos especias, que contem seus registos, a gerencia de Entrada e Saída ocorre através destes arquivo, todas as operações de entrada e saída são realizadas como uma sequencia de bytes, sendo assim é possível se comunicar com mais de um dispositivo por vez. A parte de buffer e controle de acesso ficam por conta do kernel, para manter um equilíbrio entre o kernel e os drives o sistema implementa uma interface que padroniza o acesso por parte tanto do kernel (DKI – Driver Kernel Interface) quanto do Driver (DDI – Device Driver Interface).

O DDI tem a função de isolar o dispositivo de Entrada e Saída do kernel, tornando-o independente, e somente após ser acoplado ao sistema é que o kernel é gerado, os sistemas UNIX necessitam de um reinicialização do sistema para dar funcionalidade aos novos hardwares.

Os dispositivos que transmitem um grande bloco de informações, recebem uma atenção especial do sistema operacional, para evitar demoras com busca de informações no dispositivo e transferência para a memoria, para isto é utilizado um buffer na memoria principal onde os blocos recebidos recentemente do dispositivo ficam armazenados por um determinado período de tempo. Assim quando ocorrer uma nova operação de leitura primeiramente é verificado o buffer para ver se lá não se encontra a informação necessária antes de fazer acesso ao dispositivo propriamente dito.

7.Processos e Threads

Na inicialização do sistema um processo denominado “0” é iniciado e por sua vez inicia o processo “1” chamado de init ele é responsável pro iniciar os processos posteriores, ou seja sempre que um usuário logar é ele quem inicia o shell para o novo usuário, após a inicialização do shell os próximos processos criados pelo usuário são executados pelo próprio shell ou o shell cria um processo para executa-los.

Cada comando interpretado pelo Shell gera um processo com um PID, a partir da criação deste a *system call fork,* cria os sub-processos, ou processos filhos, e os associa ao processo pai, o Sistema utiliza os PID's para acompanhar o status de cada processo.

Os status dos processos podem ser:

- Executando – Processo carregado na memoria e em execução.
- Bloqueado – Processo bloqueado pelo sistema operacional ou pelos dispositivos externos.
- Em Espera – Aguardando a vez para fazer uso do processador ou dos dispositivos externos.
- Concluído – Processo já executado.

Caso um processo seja muito grande e seja necessário executar outro processo, é possível deixa-lo para executar em *background,* ou segundo plano,

para iniciar ou transferir um processo para background basta somente colocar um “&” ao fim da linha de comando.

Os processos pertinentes ao próprio sistema são denominado de daemons, eles são responsáveis por tarefas como, gerenciamento de logs, escalonamento de tarefas , gerencias de filas de impressão, enfim por tarefas administrativas. UNIX implementa o uso de sinais que nada mais são do que avisos aos processos sobre possíveis erros em sua execução, os processos por sua vez podem aceitar estes sinais e iniciarem um tratamento especifico ou simplesmente ignora-lo e seguir sua execução.

O sistema UNIX também implementa um vetor de processos, um tipo de tabela onde ficam representadas a estrutura dos processos em execução, este vetor tem um tamanho definido que limita a quantidade de processos que podem estar executando, quando um FORK é executado o sistema busca no vetor de processos um espaço livre para criar o processo filho, só então é que são copiadas as informações do processo pai.

Inicialmente os sistemas UNIX não utilizavam Threads, mas com o crescimento de processos em execução não havia outra saída. Em 1995 é que as Threads foram implantadas no sistema. Foi criado então o padrão POSIX que definia a sincronização entre as threads como semáforos e variáveis condicionais.

Referencias

[1]. Arquitetura de Sistemas Operacionais 4ª Edição Francis Berenger Machado Luiz Paulo Maia, 2008.

[2]. Sistemas Operacionais - Vol. 11: Série Livros Didáticos Informática UFRGS Rômulo S. Oliveira, Alexandre S. Carissimi, Simão S. Toscani.

[3]. Introdução ao Sistema Operacional Unix - artigo- Cristiana Munhoz Eugênio, Daniela Regina Barbetti, 1997.

[4]. Introdução aos Sistemas Operacionais – Unicamp, Eleri Cardozo, Mauricio F. Magalhaes, 2002 (Artigo).