FACULDADE DE TECNOLOGIA SENAC GOIÁS
Gestão em Tecnologia da Informação

CentOS

Lucília Ribeiro

GOIÂNIA,
2015

RESUMO

Neste guia seguira noções especificas de um dos sistemas operacionais mais difundidos no mundo o CentOS, com particularidades abrangentes principalmente no campo da tecnologia voltada em distribuição de redes de computadores interligadas. Veremos suas principais funcionalidades, usabilidades e seu funcionamento com seus requisitos próprios. O CentOS se tornou um dos sistemas mais estáveis e robustos para implantação de aplicações e serviços.

Sumário

1 Introdução

O CentOS (Community Enterprise Operating System) é uma versão gratuita da distribuição RHEL (Read Hat Enterprise Linux), desenvolvido e mantido pelo CentOS Project, suas versões são compiladas a partir do código fonte fornecido pela Read Hat. O CentOS proporciona um grande acesso aos softwares padrão da indústria, incluindo total compatibilidade com os pacotes de softwares feitos para os sistemas da RHEL. Isso proporciona ao CentOS o mesmo nível de segurança e suporte, através de updates, que outras soluções Linux Enterprise, porém sem custo.

Uma das principais características dessa distribuição é a sua estabilidade, as versões do CentOS são mantidas por dez anos com constantes atualizações para os pacotes instalados.

O CentOS usa o yum para gerenciar seus pacotes, possuindo também o SELinux (Security-Enhanced Linux), que aumenta a segurança do sistema.

Por default ele oferece as interfaces gráficas GNOME e KDE, sendo possível instalar outras mais leves como a MATE ou XFCE, e a numeração das suas versões é baseada na numeração do RHEL.

O CentOS é usado em todo mundo por pessoas que procuram um sistema estável e robusto para implantação de aplicações e serviços.

A escolha desse sistema, para à apresentação da proposta técnica, foi feita com base nos projetos desenvolvidos nas outras matérias que exigiram um sistema com suporte a servidores.

2 Linux

O Linux é um sistema multiusuário e multitarefa com um conjunto completo de ferramentas compatíveis com o sistema Unix.

É constituído por três partes principais de código, são elas:

Kernel: O kernel é responsável por manter todas as abstrações importantes do sistema operacional, incluindo memória virtual e processos.

Bibliotecas do sistema: As bibliotecas do sistema definem um conjunto-padrão de funções através das quais as aplicações podem interagir com o kernel.

Utilitários do sistema: Os utilitários do sistema são programas que executam tarefas de gerenciamento individuais e especializadas.

O kernel do Linux forma o núcleo do sistema, ele fornece as funcionalidades necessárias para a execução de processos e serviços do sistema que concedem acesso aos recursos de hardware. As bibliotecas do sistema permitem que as aplicações façam solicitações de serviços ao kernel. Elas são encarregadas de coletar os argumentos da chamada de sistema e organiza-los de forma que a chamada possa ser realizada. As bibliotecas também fornecem rotinas que não correspondem as chamadas do sistema, além de que, todas as funções necessárias para suportar a execução de aplicações UNIX e POSIX são implementadas nas bibliotecas do sistema. Os utilitários do sistemas incluem todos os programas necessários a inicialização do sistema, podendo também executar funções de processamento de texto.

2.1 Gerência de Processos

O Linux utiliza um modelo de processos semelhante ao do UNIX, no entanto opera de forma diferente em alguns aspectos. Tanto no Linux quanto no UNIX, os processos são gerenciados de duas formas distintas: a criação de um novo processo se dá por meio do fork(), e a execução de um programa é precedido por um exec().

A criação de um processo acontece da seguinte forma: um processo existente faz uma cópia de si mesmo por meio do fork, esse novo processo é chamado de processo filho, e recebe o código que deve ser executado pela chamada exec. Todo processo criado recebe um identificador PID, único e que só pode ser reutilizado quando o processo a que ele se refere finalizar. Alguns processos são criados pelo próprio sistema operacional, como o processo criado durante a sua inicialização. Esse processo é chamado de init e por ser o primeiro processo o seu PID é 1. A partir deste processo são criados vários outros processos responsáveis pelo gerenciamento do sistema, por meio do fork-exec. Cada processo criado é associado a um descritor de processos, a criação de um novo processo atualiza todos os campos do descritor que informa o estado do processo criado.

Depois da criação de um processo, este está pronto para ser executado. Durante a fase de execução, um processo pode assumir diferentes estados. São eles:

TASK_RUNNING: Momento em que o processo está executando ou esperando para ser executado. Não há uma diferenciação entre pronto e executando. O Linux utiliza um apontador para diferenciar o processo que está sendo executado pelo processador naquele momento.

TASK_INTERRUPTIBLE: Momento em que o processo se encontra bloqueado. Neste estado ele pode estar usando ou esperando para usar um outro recurso. Após este estado o processo passa para o estado de TASK_RUNNING.

TASK_UNINTERRUPTIBLE: Processo está no estado de bloqueado, mas diferente do TASK_INTERRUPTIBLE, neste estado o processo está esperando por uma condição critica.

TASK_STOPPED: Momento em que o processo é parado por uma interrupção de software. O processo só volta a ser executado quando recebe outra interrupção de software.

TASK_ZOMBIE: Estado que um processo filho assume ao terminar enquanto espera a execução por parte do processo pai. Após a execução do processo pai o processo filho é liberado.

Quando um processo é finalizado, ele sinaliza o seu termino por meio da chamada exit(). Essa chamada é feita automaticamente sempre que o último comando do programa é executado.

O Thread é um fluxo de controle no interior de um processo, ele é implementado por meio de uma chamada clone no sistema. Um processo clone compartilha recursos com o processo original. Quando dois ou mais processos compartilham a mesma estrutura, eles atuam como se fossem diferentes threads no interior de um só processo. O Linux não faz distinção entre processo e thread, referindo-se a um thread ou um processo como uma tarefa, quando se refere a um fluxo de controle dentro de um programa. Isso é possível porque o Linux não mantem o contexto inteiro de um processo dentro da estrutura de dados do

processo principal, em vez disso, ele mantem o contexto dentro de subcontextos independentes. A estrutura de dados do processo contém os ponteiros para essa estrutura, permitindo que, qualquer número de processos possa compartilhar um subcontextos apontando para o mesmo subcontextos.

Quando a chamada clone é invocada, ela recebe um conjunto de flags que determinam o nível de compartilhamento que deve ocorrer entre o processo pai e o processo filho. Caso nenhuma dessas flags sejam passadas quando o clone for invocado, não ocorrerá compartilhamento, resultando em funcionalidade semelhante à da chamada de sistema fork.

A vantagem de criar uma thread está em seu custo de criação, elas são criadas mais rapidamente que processos pois não há necessidade de copiar os atributos do processo original, basta referenciar com os ponteiros as áreas já existentes do processo que está sendo clonado.

2.2 Gerenciamento de Memória

O gerenciamento de memória no Linux se dá de duas formas diferentes, uma trata da alocação e liberação de memória física como páginas, grupo de páginas e blocos de memórias; outra trata da manipulação da memória virtual.

Usualmente, o Linux separa a memória física em três zonas distintas: ZONE_DMA; ZONE_NORMAL; ZONE_HIGHMEN.

O tamanho de cada zona depende da arquitetura utilizada. A zona DMA contém a localização da memória principal, que inclui as localizações de memória de 0 MB a 16 MB. Essa zona existe apenas quando a arquitetura limita o que o DMA pode acessar, do contrário a zona NORMAL é utilizada em seu lugar, ela inclui as localizações de memória de 16 MB a 896 MB. A zona HIGHMEN, ou memoria alta, refere-se a memória física que não é mapeada no espaço de endereçamento do kernel.

O Linux possui um alocador de páginas responsável por alocar e liberar todas as páginas físicas para a zona, alocando intervalos de páginas quando necessário. Esse algoritmo de alocação é chamado de buddy. Ele faz uso do sistema de pares que é utilizado para procurar páginas físicas disponíveis que tenham o tamanho requisitado. Sempre que duas regiões parceiras são liberadas, elas são unidas formando uma região maior. Essa região maior também tem uma região parceira que pode se juntar a ela formando outra região maior que a anteriormente existente. De forma similar, quando uma solicitação demanda uma região maior que a região disponível, uma região maior existente é alocada e dividida. A alocação de memória pode ser feita estaticamente, quando o sistema, ao ser inicializado reserva uma área de memória, ou dinamicamente, pelo alocador buddy.

Alguns subsistemas de gerencia de memória não utilizam o alocador de páginas para gerenciar seu pool de memória, os mais importantes desses sistemas são: o kmalloc, o alocador de placas, o cache de páginas e o sistemas de memória virtual.

O serviço *kmalloc()* aloca páginas inteiras sob demanda, e em seguida, divide-as em pedaços menores. As páginas alocadas pelo kmalloc são mantidas pelo kernel em listas, essas regiões são alocadas permanentemente e a sua liberação deve ser feita de forma explicita.

O alocador de placas é utilizado para alocar memória para a estrutura de dados do kernel. Esse algoritmo utiliza caches para armazenar os objetos do kernel. Uma placa pode assumir três estados diferentes: cheia – todos os objetos contidos na placa estão sendo utilizados; vazia – todos os objetos contidos na placa estão livres; parcial – dos objetos contidos na placa alguns estão livres e outros estão sendo utilizados. Nesse subsistema, o alocador de placas procura atender à solicitação utilizando um objeto livre de uma placa no estado parcial. Caso não exista uma placa em estado parcial, é utilizado um objeto a partir de uma placa em estado vazio. No caso de não existir uma placa em estado vazio disponível, uma nova placa é alocada a partir das páginas físicas e atribuída a um cache.

O cache de página é projetado para diminuir o tempo gasto executando operações de E/S. Ele armazena páginas de conteúdo de arquivos, podendo também armazenar dados de rede em cache. O mapeamento das páginas no cache de páginas para a leitura de uma página de dados no cache, é feita utilizando o sistema de memória virtual. As páginas são divididas em dois grupos: páginas ativas e páginas inativas. Uma página é ativa quando foi referenciada recentemente. O Linux utiliza uma variação do algoritmo do relógio para a substituição das páginas. Esse algoritmo assegura que páginas recentemente referidas não sejam selecionadas para substituição. Dessa forma, uma página que esteja na lista ativa não poderá ser selecionada para substituição

O sistema de memória virtual é responsável por manter visível o espaço de endereçamento de cada processo. Ele cria páginas na memória de acordo com a necessidade, e seu gerenciamento se dá a partir do disco. O Linux visualiza a memória virtual de duas formas: como um conjunto de regiões separadas e como um conjunto de páginas. A visão de um conjunto de regiões separadas é a visão logica, e descreve as instruções relacionadas ao formato do espaço do endereçamento que o sistema recebeu. Já a visão de um conjunto de páginas é a visão física e é armazenada nas tabelas de página de hardware do processo.

A gerencia de memória no Linux é feita com base na paginação, a memória virtual de um processo é dividida em páginas e um problema que poderia ocorrer com esse tipo de organização é resolvido por meio da paginação por demanda. Na paginação por demanda, o sistema procura manter em memória somente as partes do código que estão sendo utilizadas. Dessa forma, o núcleo do Linux mantem apenas uma estrutura que descreve a memória virtual do processo indicando quais partes estão carregadas na memória e quais estão no disco. Assim, sempre que um processo precisa acessar uma página que não está na memória é gerada uma falta de página, essa falta de página é tratada pelo sistema da seguinte forma: a página que está sendo demandada é carregada na memória, caso não haja espaço disponível, o sistema de gerencia de memória realiza o swapping, que substitui uma página. O Linux utiliza um sistema de paginação em três níveis para traduzir o endereço virtual: páginas lógicas, para endereços reais, e páginas físicas, também chamadas de frames.

2.3 Sistema de Arquivos

As versões atuais do Linux tem suporte a grande parte dos sistemas de arquivos existentes. O ext2fs, por exemplo, é o sistema de arquivo padrão, por razões históricas, o principal objetivo desse sistema de arquivos é fornecer um alto desempenho com suporte a

recursos avançados. Cada sistema de arquivos determina como armazenar e acessar seus dados. O Linux herda um conceito muito importante do UNIX que é o ponto de montagem. O ponto de montagem é o diretório "/" que corresponde a raiz que define o sistema de arquivos. Partindo desse diretório, é possível montar outros sistemas de arquivos.

O Linux visualiza todos os sistemas de arquivos da perspectiva de um conjunto de objetos comum. Esses objetos são superblock, inode, dentry e file. Na raiz de cada sistema de arquivos está superblock, que descreve e mantém o estado do sistema de arquivos. Cada objeto que é gerenciado dentro de um sistema de arquivos (arquivo ou diretório) é representado no Linux como um inode. O inode contém todos as operações para gerenciar objetos no sistema de arquivos. O conjunto de estruturas, chamadas dentries, é usado para conversão entre nomes e inodes, para o qual um cache de diretório existe para manter por perto o objeto usado mais recentemente. O dentry também mantém relacionamentos entre diretórios e arquivos para sistemas de arquivos desviados. Por fim, um arquivo VFS representa um arquivo aberto. O VFS atua como o nível raiz da interface do sistema de arquivos. Ele mantém o rastreio dos sistemas de arquivos suportados atualmente, bem como dos sistemas de arquivos que estão sendo montados no momento. Os sistemas de arquivos podem ser incluídos ou removidos dinamicamente do Linux usando um conjunto de funções de registro. O kernel mantém uma lista dos sistemas de arquivos suportados, que pode ser visualizada a partir do espaço de usuário através do sistema de arquivos /proc.

3 CONCLUSÃO

Como vimos, o CentOS é um sistema feito e pensado para servidores, com forte poder de gerencia e de trabalhos em redes, sendo também bastante utilizado em desktops.

Por manter sua gratuidade, ele se torna uma ótima opção para ambientes estudantis e de testes. O CentOS não possui suporte técnico pago, como nas distribuições Linux Read Hat, no entanto dispõe de uma ampla comunidade, com fóruns que dão suporte, além de também possuir uma extensa documentação.

Outro ponto forte deste sistema é sua estabilidade, suas versões são mantidas por até dez anos com constantes atualizações, o que lhe proporcionou vários usuários.

Contudo, o CentOS pode ser considerado um ótimo sistema para implantação de servidores. Fácil de usar, intuitivo e acima de tudo seguro, o CentOS cumpre o que promete e mesmo usuários com pouca experiência em sistemas Linux não terão dificuldades em utiliza-lo, e obter informações pertinentes sobre essa ótimo sistema operacional.

4 Referências Bibliográficas

DEITEL, H.M. **Sistemas Operacionais.** São Paulo: Pearsoned, 2005.

SILBERSCHATZ, Abraham. **Fundamentos de Sistemas Operacionais.** Rio de Janeiro: LTC, 2013.

OLIVEIRA, Rômulo Silva de. **Sistemas Operacionais.** Porto Alegre: Bookman, 2010.

MEMBREY, Peter; VERHOEVEN, Tim; ANGENENDT, Ralph. **The Definitive Guide to CentOS.** New York: Apress, 2009. Disponível em:<http://www.e-reading.club/bookreader.php/137962/Angenendt_-_The_Definitive_Guide_to_CentOS.pdf> Acesso em: 24 nov. 2015.