Projeto 1 - Introdução ao Minix

Giovanna Vendramini (173304), Luiz Eduardo T. C. Cartolano (183012) e Rafael Figueiredo Prudencio (186145)

I. Resumo

O objetivo do projeto é que nós nos familiarizemos com a plataforma do MINIX, de modo a começarmos a entender sua estrutura e funcionamento.

Na primeira parte do projeto, focamos nossos esforços na instalação e setup geral do sistema operacional em uma virtual machine, uma tarefa bem simples e rápida, a qual obtivemos sucesso.

Uma vez instalado e funcionando, começamos de fato a trabalhar em alterações no sistema. A primeira delas foi fazer com que, a cada comando executado no terminal, fosse exibida a mensagem: *printf("executando: %s\n", fullpath);*, cujo resultado pode ser observado na Figura 1. A segunda alteração feita, foi a modificação do banner de inicialização do *kernel*, de modo que o mesmo exibisse uma mensagem personalizada, como nos mostra a Figura 2.

Por fim, na última atividade proposta, nos concentramos em descobrir qual seria a *Syscall 33* do MINIX, que, através do estudo dos livros de referência sobre o sistema, descobrimos ser a chamada PM_CLOCK_GETRES.

II. Introducao

O MINIX é um sistema operacional baseado em UNIX que foi criado com propósitos educacionais por Andrew S. Tanenbaum. O primeiro release do MINIX aconteceu em 1987 e desde então é um software livre, com seu código fonte aberto para estudantes e pesquisadores do mundo todo. As versões 1 e 2 do MINIX foram criadas como ferramentas de ensino, enquanto o MINIX 3, utilizado nesta disciplina, tem como novo objetivo ser usado como um sistema de alta confiança em sistemas embarcados e computadores com recursos limitados.

O MINIX é baseado em um pequeno microkernel rodando no modo kernel com o resto do sistema operacional rodando como inúmeros processos isolados e protegidos em modo usuário. O design de microkernel do MINIX torna-o uma opção muito melhor que o Linux para sistemas embarcados, nos quais às vezes é fundamental que uma aplicação rode por tempo indeterminado. O kernel monolítico do Linux implica em uma necessidade de reiniciar o sistema operacional após qualquer update no sistema de arquivos ou drivers de dispositivos. Um sistema capaz de atualizar-se e reparar-se sem a necessidade de um reboot pode ser muito valioso para algumas aplicações, como por exemplo, em usinas elétricas.

III. Tarefas

A. Instalação

Já havendo trabalhado com o VirtualBox da Oracle, optou-se por seguir as instruções especificadas em [1] para configurar o MINIX 3 em uma máquina virtual. Preferiu-se baixar a ISO mais recente da página de development snapshots, no caso, a versão *3.4.0rc6* do MINIX. Em seguida, foi necessário criar uma nova máquina virtual pelo VirtualBox. As especificações da máquina não precisam ser muito pesadas, já que o MINIX foi criado justamente para ser um sistema operacional leve. Seguindo as recomendações em [1], dedicou-se 1GB de memória 8GB de espaço de armazenamento para a máquina virtual. Após montar a imagem de disco no VirtualBox, basta clicar na nova máquina virtual criada para começar a instalação do MINIX, na qual foram selecionadas todas as opções default. Para bootar pelo sistema recém instalado, basta desmontar a imagem da máquina virtual.

Em seguida, foram feitas as configurações do MINIX, inicialmente com o comando *setup*. Depois foi adicionada uma senha ao usuário root ao usar o comando *passwd* e foi configurado o horário de acordo com o do estado de São Paulo, com o comando

echo export TZ=America/SaoPaulo>/etc/rc.timezone.

O próximo passo foi criar e popular o banco de dados inicial dos pacotes através do comando (*pkgin update*). Depois instalamos o OpenSSH para obter o client (ssh) e o daemon (sshd) através do comando *pkgin install openssh*. Em seguida, foi necessário configurar o daemon para rodar na máquina, copiando seu código fonte para a pasta de */etc/rc.d*, diretório onde habitam os "run commands". Adicionando a linha *sshd=Yes* ao arquivo */etc/rc.conf* está completa a configuração. Agora basta permitir acesso root ao daemon modificando o arquivo */usr/pkg/etc/ssh/sshd_config*. Por fim, através do comando *pkgin_sets*, instalou-se cerca de 700 pacotes comumente usados.

B. Mudança do exec

Seguindo a indicação do enunciado de modificar a syscall *exec*, buscou-se por arquivos que possam estar relacionados a essa chamada. Primeiro, executou-se o comando *find /usr/src/minix -type f -name "*exec*"*, permitindo-nos localizar os arquivos relacionados à chamada *exec*. A partir do resultado da busca, modificou-se os arquivos considerados pertinentes até encontrar um que imprimisse a saída esperada. Experimentou-se modificar o arquivo *minix/kernel/system/do_exec.c*, mas o argumento *name* passado dentro de um ponteiro de struct consistia apenas do nome do programa e não de seu caminho completo no sistema de arquivos.

Finalmente, a mudança desejada foi possível ao modificar o arquivo *minix/servers/vfs/exec.c*, onde o argumento *fullpath* do tipo *char** da função *get_read_vp* correspondia à string desejada. Adicionando a linha *printf("executando: %s\n", fullpath);* ao início da função obteve-se a saída indicada pela Figura 1.

Vale notar que um grande motivo de confusão foi a presença de múltiplos arquivos com nomes semelhantes, como o *minix/servers/pm/exec.c*. A tarefa foi realizada através de

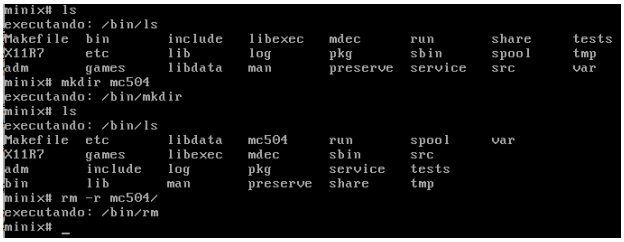

Fig. 1. Saída no terminal do MINIX após modificar o kernel para imprimir o caminho do arquivo que está executando.

Fig. 2. Saída no terminal do MINIX após modificação do banner do kernel.

tentativa e erro, mas em retrospectiva, é claro o motivo pelo qual as outras mudanças não funcionaram. O arquivo pertencente ao process manager implementa funções intermediárias como *do_exec* que repassa a syscall para o sistema virtual de arquivos, *do_newexec* que toma conta dos setuid bits e *exec_restart* que termina uma syscall exec regular. O arquivo possui como pontos de entrada estágios intermediários da execução de um processo, mas nenhum deles requer como argumento o caminho completo para o binário do processo que deseja-se executar. Esse caminho é manipulado pelo sistema virtual de arquivos para criar o executável a partir do binário.

C. Banner do kernel

A segunda atividade de mudança solicitada para o projeto era a de modificar o banner do *kernel*, alterando a frase inicial ("Minix 3.4.0 Copyright...") desse.

Primeiro, de maneira similar a feita para modificar a syscall *exec*, executou-se um comando para buscar os arquivos no sistema que continham uma das linhas impressas no banner. Através do comando

grep -r "http://www.minix3.org" minix/

reduziu-se o espaço de busca para dois arquivos: *minix/kernel/main.c* e *minix/servers/vsf/README*. Sendo um deles um arquivo de documentação, restou apenas a primeira opção. Dentro do arquivo *main.c*, existe a função *static void announce(void)* que, segundo a documentação do sistema, é a responsável pelo display do banner de inicialização do *kernel* do Minix.

Uma vez encontrado o que deveríamos modificar, bastou inserir comandos de *printf* após a linha *printf(""Minix is open source software ... ")*, obtendo-se o resultado dado pela Figura 2.

D. Syscall 33

No contexto dos processos rodando em um sistema operacional, um servidor é um processo que oferece serviços úteis para os processos do usuário. De acordo com [2], dois servidores são essenciais em um sistema operacional: o gerenciador de processos (PM do inglês process manager) e o sistema virtual de arquivos (VFS do inglês virtual file system). O PM está encarregado de fazer todas as chamadas de sistema que envolvem inicializar ou interromper a execução de um processo, como *fork*, *exec* e *exit*, além das chamadas de sistema relacionadas a upcalls como *alarm* e *kill*. O VFS faz todas as chamadas de sistema relacionadas a arquivos como *read*, *mount* e *chdir*.

Para poder gerenciar as chamadas, tanto o PM como o VFS possuem um arquivo que instancia uma tabela mapeando o número da chamada (a posição no array) com a rotina em C responsável por executá-la. No diretório *servers/pm* e *servers/vfs* do código fonte do MINIX encontrou-se o arquivo C chamado *table.c* responsável por esse mapeamento. Avaliando esse arquivo, notou-se que as constantes usadas para indicar a posição no vetor estavam declaradas no arquivo *callnr.h*. Inspecionando o arquivo *callnr.h*, localizado no diretório *minix/include/minix*, encontrou-se as definições que associam o nome das chamadas ao número delas. Na linha 46 do arquivo temos:

#define PM_CLOCK_GETRES (PM_BASE + 33)

Como PM_BASE foi inicializado como 0x000, conclui-se que a syscall 33 refere-se a chamada CLOCK_GETRES, que retorna a precisão do relógio do sistema.

IV. Conclusão

Foi possível completar todas as tarefas que nos propusemos a realizar. A instalação do MINIX foi trivial. Foi necessário apenas seguir as instruções disponíveis em [1]. A mudança da syscall *exec* foi a tarefa mais trabalhosa, já que haviam muitos arquivos possíveis para modificar e devido a pouca familiaridade com o sistema não estava claro o motivo pelo qual algumas modificações não funcionavam. No final, a mudança no arquivo *minix/servers/vfs/exec.c* teve o resultado desejado. A modificação do banner também foi trivial e foi feita através de uma busca rápida pelos arquivos que continham o texto original do banner. Finalmente, a syscall 33 causou uma pequena confusão devido aos dois servidores que faziam chamadas de sistema: PM e VFS. Investigando os valores das constantes no arquivo *minix/include/minix/callnr.h* concluiu-se que trata-se da chamada CLOCK_GETRES.

References

[1] Minix 3 wiki. Accessed 16-03-2018. [Online]. Available: https://wiki.minix3.org

[2] A. S. Tanenbaum, *Modern Operating Systems*, 3rd ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2007.