UNIVERSIDADE FEDERAL DE SÃO CARLOS

Bacharelado em Ciência da Computação
Laboratório de Sistemas Operacionais
Professor Gustavo Maciel Dias Vieira
***Campus* Sorocaba**

Projeto 3
Módulos e Estruturas Internas do Núcleo

Rafael Paschoal Giordano - 408298
Thales Gonçalves Chagas - 408557

Sorocaba

2014

Introdução

O projeto 3 têm como objetivo aprender como expandir a funcionalidade do núcleo com a criação de módulos personalizados assim como explorar e alterar as estruturas internas que guardam informações sobre processos.

Descrição

O projeto consiste em três tarefas:

1. Compilar o módulo fornecido como exemplo e verificar seu funcionamento.
2. Modificar o módulo fornecido para exibir o PID do processo lendo o arquivo e o PID do processo pai.
3. Modificar o módulo fornecido para dar ao interpretador de comandos executando o processo de leitura permissões de root.

Segue as explicações:

1. Compilar o módulo fornecido como exemplo e verificar seu funcionamento.

   O módulo fornecido foi compilado e com isso foi criado um arquivo hello.ko, este foi inserido na memória para ser testado através do comando insmod. Ao executar cat /proc/hello, a mensagem "Hello, World" foi apresentada mostrando assim que o módulo foi carregado e executado com sucesso. Então remove o módulo, que era apenas um teste com rmmod.

2. Modificar o módulo fornecido para exibir o PID do processo lendo o arquivo e o PID do processo pai.

   O núcleo do sistema operacional usa uma estrutura própria denominada task_struct para guardar os dados de processos, a qual possui características próprias para ser utilizada. Para ter acesso ao PID dos processos foi utilizada as estruturas presentes na biblioteca sched.h, presente nos arquivos de código fonte do núcleo. Com a execução do módulo é utilizada a macro *current* que se mostra como um ponteiro para o registro do processo atual. Com o uso desta é possível acessar e mostrar na tela os PIDs dos processos pai e filho, sendo o do filho o atual, e o do pai usando *current->parent que* aponta para o processo pai do atual. É utilizada a função seq_printf para a conversão e impressão dos números. Mostrado o código na imagem abaixo:

```
seq_printf(m, "PID filho:%d\n",current->pid);

seq_printf(m, "PID pai:%d\n",current->parent->pid);
```

   Usa função para imprimir primeiro PID do processo atual, e então PID do seu pai.

3. Modificar o módulo fornecido para dar ao interpretador de comandos executando o processo de leitura permissões de root.

   Nesta tarefa foi utilizada a biblioteca cred.h que é responsável por definir as credenciais dos processos. As credenciais podem ser encontradas na estrutura *cred* presente nesta biblioteca, mas para acessá-las é preciso fazer uma requisição e depois

atualizar as alterações feitas, para isso são utilizados um ponteiro do tipo struct cred e as funções que fazem a requisição e posteriormente a atualização das credenciais do processo, get_cred e put_cred, respectivamente. Após obter as credenciais do processo a ser modificado, as do interpretador, ou seja, processo pai, as alterações devidas para as credenciais do processo serem elevadas são feitas ao fazer as variáveis receberem id *GLOBAL_ROOT_UID*. Após essas alterações, tínhamos privilégio de root para a as execuções relacionadas ao módulo criado. Mostrado o código na imagem abaixo:

```
ponteiro_pid = (struct cred *) get_cred(current->parent->cred);

ponteiro_pid->uid = GLOBAL_ROOT_UID;
ponteiro_pid->gid = GLOBAL_ROOT_GID;
ponteiro_pid->euid = GLOBAL_ROOT_UID;
ponteiro_pid->egid = GLOBAL_ROOT_GID;

put_cred(ponteiro_pid);
```

Imagem mostra código para alteração de credenciais do processo.

Dificuldades

As principais dificuldades foram entender como usar as estruturas próprias para processos do sistema operacional, e como trabalhar com elas. Mas após algum tempo trabalhando sobre o projeto estas se mostraram bastante úteis e de boa relevância para a implementação, assim como as funções auxiliares utilizadas.

Conclusão

Com o estudo e entendimento das estruturas de dados internas ao núcleo, se torna possível a criação de módulos que venham a ser úteis ao sistema operacional, tornando este mais flexível quanto aos módulos.