Projeto 2 - MC504

Universidade Estuadual de Campinas

1º Semestre de 2020

Victor Rigatto - RA 178068

1 Resumo

Este trabalho tem como objetivo principal implementar um novo serviço ao sistema operacional MINIX, para resolver um problema de sincronização através de mutex e variáveis de condição. Por se tratar de um microkernel, a maioria das funcionalidades do Minix são implementadas através de serviços que se comunicam por mensagens e, originalmente, o sistema não possui a capacidade de sincronizar processos simultaneamente. A implementação do serviço de mutex e variáveis de sincronização providenciam essa funcionalidade.

2 Introdução

MINIX, abreviacao para Mini UNIX, e um sistema operacional livre e de codigo aberto, e tem sua arquitetura baseada em um microkernel. Este tipo de sistema operacional se constitui basicamente de um kernel pequeno, rodando em kernel-mode, enquanto o restante do sistema roda através de um conjunto de processos (serviços), isolados e protegidos em user-mode. O microkernel lida com as interrupções, providencia os mecanismos básicos para gerenciamento de processos, implementa a comunicação entre processos, assism como gerencia o scheduling dos processos. Por outro lado, o sistema de arquivos, gerenciamento de processos, rede e outros serviços são disponibilizados por servidores separados, fora do microkernel, como serviços. O microkernel suporta algumas chamadas de sistema, denominadas system-tasks. Outras chamadas de sistema, utilizadas pelos serviços descritos anteriormente, são processadas fora do microkernel, ou seja, os servidores cuidam de chamadas de sistema.

Este trabalho consiste em implementar um novo serviço, servidor, ao sistema operacional do MINIX. Este serviço será responsável por providenciar a capacidade de sincronização de processos simultaneamente, através da implementação de um serviço mutex e variáveis de sincronização. Mutex é uma forma de implementação simplificada de semáforo, que permite o gerenciamento de exclusão mútua de recursos compartilhados, capacidade necessária para a implementação de threads. O funcionamento do mutex será melhor descrito posteriormente.

A criação de um novo serviço se dá pela implementação de um novo servidor, que funciona como um system call handler, ou seja, um serviço que trata chamadas ao sistema específicas, a criação de novos tipos de mensagens para o novo serviço, que são a forma nativa de comunicação inter processos (IPC) do sistema operacional MINIX, e a criação ou modificação de uma biblioteca de usuário, responsável por direcionar as chamadas de sistema para os servidores apropriados, assim como seus argumentos pertinentes e potencialmente retornar um resultado.

3 Desenvolvimento

3.1 Modificações e a estrutura de dados e arquivos

A estrutura de dados de arquivos que contém os arquivos fonte pertinentes à implementação de servidores, mensagens e bibliotecas estão definidos a seguir. Como a implementação do novo serviço está fora do microkernel, a compilação pode ser feita diretamente no diretório dos arquivos com os códigos fontes pertinentes, utilizando-se do Makefile local para esses arquivos. No entanto, algumas modificações requerem a recompilação total do sistema para entrar em vigor, como é o caso de arquivos para a inicialização.

O código fonte para os servidores se encontra no diretório /usr/src/minix/servers. Cada servidor possui o seu próprio diretório, e podemos compilar os servidores através do Makefile desse diretório. O novo servidor foi implementado em um novo diretório, em /usr/src/minix/servers/syncvar, como sugerido pelo ennciado do projeto. No novo diretório estão os arquivos fontes para os novos system call handlers do serviço de sincronização simultânea, implementados em /main.c (contém as rotinas de inicialização e encerramento do servidor), mutex.c (contém a implementação do mutex), cond.c (contém a implementação da variável de condição), svar.h (contém a implementação da variável de sincronização), queue.c (contém a implementação da fila para o mutex) e queue.h. (contém o cabeçalho com a estrutura de dados para a fila). O Makefile para o novo servidore, em /usr/src/minix/servers/syncvar, também foi criado para incluir o novo servidor.

Para a inicialização dos servidores na inicialização do sistema, foi definido o novo servidor em /usr/src/etc/defaults/minix.rc.conf (arquivo de configuração de inicialização do MINIX) e /usr/src/etc/rc.d (contém o executável para syncvar).

O novo serviço foi adicionado em /usr/src/etc/system.conf (arquivo de configuração de serviços na imagem de boot).

A implementação do corpo da função responsável por tratar as novas chamadas ao sistema e direcioná-las ao serviço pertinente através da comunicação entre processos (IPC) foi realizada em /usr/src/minix/lib/libc/sys/syncvar.c. O Makefile relativo também foi atualizado para que fosse compilado o novo arquivo fonte corretamente.

As constantes das chamadas de sistema e a estrutura de dados de mensagens de comunicação entre processos para o novo serviço foram definidas em arquivos de cabeçalho em /usr/src/minix/include/minix/com.h e em /usr/src/minix/include/minix/ipc.h.

Os prototypes para as novas chamadas ao sistema foram adicionados no arquivo de function prototypes para a biblioteca do sistema, em /usr/src/minix/kernel/system.h.

A definição dos erros para o novo serviço foram implementadas em /usr/src/minix/sys/sys/errno.h (arquivo de definição de constantes de erro).

3.2 Mutex

Mutex é uma forma de implementação simplificada de semáforo, que permite o gerenciamento de exclusão mútua de recursos compartilhados, capacidade necessária para a implementação de threads. Funcionam da seguinte maneira: se um processo tenta realizar um lock que já está com outro processo, esse processo é colocado para dormir. Quando esse lock for liberado, o processo poderá obter o lock. Esse processo se repete caso outros processos tentem adquirir um lock ocupado, criando assim uma fila. Em outras palavras, é uma forma de proteger dados compartilhados acessados por múltiplos processos, utilizando-se de exclusão mútua e modificações atômicas. A implementação se da definindo em uma fila FIFO de cada objeto que está esperando por um mutex. Os dados são acessados pelas estruturas.

mutex init(): função que cria a mensagem MUTEX INIT para ser interpretada pelo novo servidor. Essa chamada cria um mutex inicialmente livre e ativo.

mutex destroy(int mutex num): função que cria a mensagem MUTEX DESTROY para ser interpretada pelo novo servidor. Essa chamada libera um mutex (mutex num) ativo que não está bloqueado, e o torna inativo.

mutex lock(int mutex num): função que cria a mensagem MUTEX LOCK para ser interpretada pelo novo servidor. Essa chamada obtém um lock. Caso o lock não esteja dispoível, o processo que chamou é bloqueado e colocado numa fila FIFO do mutex correspondente (mutex num).

mutex unlock(int mutex num): função que cria a mensagem MUTEX UNLOCK para ser interpretada pelo novo servidor. Essa chamada libera um lock. Caso existam processos aguardando por esse lock, o primeiro da fila FIFO é acordado.

3.3 Variáveis de condição

Variáveis de condição são objetos de sincronização para mudanças em objetos compartilhados e protegidos por um mutex.

cond init(): função que cria a mensagem COND INIT para ser interpretada pelo novo servidor. Essa chamada cria uma variável de condição inicialmente livre e ativa.

cond destroy(int cond num): função que cria a mensagem COND DESTROY para ser interpretada pelo novo servidor. Essa chamada libera uma variável de condição (cond num) que não está sendo usada, colocando-a em um estado inativo.

cond wait(int cond num, int mutex num): função que cria a mensagem COND WAIT para ser interpretada pelo novo servidor. Essa chamada libera atomicamente o mutex (mutex num) e suspende a execução do processo que a chamou, colocando o processo na fila de espera FIFO da variável de condição (cond num).

cond signal(int cond num): função que cria a mensagem COND SIGNAL para ser interpretada pelo novo servidor. Essa chamada libera um processo

bloqueado da fila FIFO correspondente (cond num).

cond broadcast(int cond num): função que cria a mensagem COND BROADCAST para ser interpretada pelo novo servidor. Essa chamada libera todos os processos bloqueados da fila FIFO correspondente (cond num).

4 Conclusão

O funcionamento geral dos serviços externos ao microkernel do MINIX se dá da seguinte forma simplificada: quando uma chamada ao sistema é requisitada, o serviço (servidor) reconhece a chamada e realiza o trabalho requisitado. A chamada é feita através de uma biblioteca, que a direciona para o serviço (servidor) correspondente junto com os parâmetros pertinentes, que então trata a chamada. Essa comunicação é feita através de IPC, um recurso de comunicação entre processos nativo para o sistema operacional MINIX. Essa comunicação é definida e numerada sistematicamente. Existe um identificador de 4 bytes de quem enviou a mensagem, um identificador de tipo de mensagem de 4 bytes, e 56 bytes para enviar ponteiros para dados.

A implementação de mutex, assim como de variável de condição, são bastante documentados. O livro texto da disciplina também mostra como implementa-los. O maior desafio foi adaptar a implementação para funcionar com os parâmetros no contexto do MINIX. O funcionamento dessas funcionalidades é descrito de forma genérica na bibliografia, porém a implementação da comunicação entre os serviços envolveu entender como esses serviços funcionam. Felizmente, foi possível ver exemplos de serviços já implementados no MINIX, para entender o sistema de comunicação. Além disso, a implementação se torna relativamente repetitiva para as funções de mutex e variáveis de condição, depois que fica mais claro como se dá a comunicação e a passagem de parâmetros no MINIX.

As funções precisam ser definidas em diversos arquivos, e essa foi uma das partes mais confusas e difíceis, devido a dificuldade de encontrar a documentação correta para essas questões. É preciso definir também as estruturas de dados para o mutex, para a variável de condição e para a mensagem. Todas essas adições foram feitas em arquivos específicos, e foi bastante difícil encontrá-los. As funções e chamadas também devem todas corresponder umas às outras, e o novo serviço precisa ser adicionado à inicialização do sistema.

Outro ponto é sobre o sistema de construção (build) do MINIX, que permite utilizar Makefile locais para construir e linkar os componentes do sistema operacional. Isso facilitou testar os novos serviços e funções localmente, permitindo checar o funcionamento de algumas funções. A compilação total do sistema é um processo mais demorado, pela natureza da máquina virtual e limitação a uniprocessador.

5 Divisão do trabalho

Os outros 3 membros do grupo 16 parecem ter desistido da disciplina. Assim, implementei sozinho o máximo que pude das funções propostas, assim como a realização deste relatório.

6 Referências

OS:P+P: Operating Systems: Principles and Practice 2ed

O Livro do MINIX: Sistemas Operacionais: P e I 3ed

https://wiki.minix3.org/

https://wiki.minix3.org/doku.php?id=developersguide:messagepassing

http://www.cis.syr.edu/ wedu/seed/Documentation/Minix3/How to add system call.pdf

https://blog.heabuh.com/jekyll/update/2016/11/13/minix-service-tutorial.html

http://www-di.inf.puc-rio.br/ endler//courses/inf1019/Minix/Minix-Syscall-tutorialv2.pdf

https://www.freebsd.org/cgi/man.cgi?query=rc.confsektion=5

https://en.wikipedia.org/

https://stackoverflow.com/