ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO

Avenida Professor Luciano Gualberto, travessa 3 nº 158 CEP 05508-900 São Paulo SP
Telefone: (11) 3091-5583 Fax (11) 3091-5294

Departamento de Engenharia de Computação e Sistemas Digitais

PCS 2042 – Sistemas Operacionais

Projeto 04: Log de utilização da memória

Autores:	Data de emissão:
Alexandre Frandulic Shimono André de Oliveira Lima Ikeda André Felipe Santos André Henrique Sanches Belloti André Kenji Horie André Romero Goncales	23/07/2007

ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO

Avenida Professor Luciano Gualberto, travessa 3 nº 158 CEP 05508-900 São Paulo SP
Telefone: (11) 3091-5583 Fax (11) 3091-5294

Departamento de Engenharia de Computação e Sistemas Digitais

Enunciado:

Mostrar através de um log como a memória foi alocada e desalocada a processos. Toda vez que um processo executar um fork, exec, exit afeta a alocação de memória e isso é registrado no log (em um deterinado arquivo). No log contém informações como:
fork - processo /usr/bin/init + segmento 0x30 - 5 clicks.
exec - processo /usr/bin/firefox + segmento 0x40 - 7 clicks.
exit - processo /usr/bin/firefox - segmento 0x40 - 7 clicks.

Objetivo:

Este trabalho tem por objetivo o estudo da organização dos processos pelo minix3, especialmente no que diz respeito à etapa de transição entre processos principais quando um novo processo é criado e executado. O log criado contendo as informações de como memória é alocada cada vez que um fork, exec e exit são executados permitem acompanhar melhor como o process manager lida com estas rotinas.

Estruturas e funções utilizadas pelo MINIX:

Para a construção deste log, foi feita uma pesquisa de como eram feitas essas system calls (fork, exec, exit). Por fim, localizamos, no diretório do Gerenciador de Processos (pm). A execução das mesmas ocorre em dois arquivos .c: **forkexit.c** e **exec.c**. Ambos os arquivos contém as funções que realizam as syscalls, no caso:

- Fork – função **do_fork** (em forkexit.c);
- Exec – função **do_exec** (em exec.c);
- Exit – função **pm_exit** (em forkexit.c).

Estas são as funções que serão alteradas. Em todas elas, estão presentes estruturas que armazenam as informações relevantes ao gerenciamento de processos:

- **rmp** – ponteiro para o processo pai
- **rmc** – ponteiro para o processo filho

Ambas são declaradas como **mproc**, que está declarado no mesmo diretório em **mproc.h**. Esta estrutura é a que contém as informações mencionadas acima. No caso, para fazermos o log de acordo com as especificações do enunciado, 3 campos nos interessam:

- nome do processo: identificado no campo **mp_name**, vetor de char
- segmento utilizado e tamanho do mesmo: são campos dentro de um campo de mproc, **mp_seg**, que é um mapa de memória, **mem_map**, que é declarado em outro arquivo, **type.h**, no diretório **/usr/src/include/minix**. Os dados que nos interessam são:
    - **mem_phys**: mostra o endereço
    - **mem_len**: mostra o tamanho do segmento

Ambos os últimos estão declarados como **clicks**, respectivamente físicos e virtuais. Estes mesmos clicks são implementados como variáveis int, no código.

Assim, estas são as estruturas que serão utilizadas para construir o log de utilização da memória.

Implementação

Para realizar a construção do log, decidimos elaborá-lo no seguinte caminho: **/var/log/memory.log**. Para se acessar este arquivo, utilizamos a função **open**, que utiliza um caminho e flags, indicadas na biblioteca **fcntl.h**. Em seguida, usamos a função **sprintf**, que funciona diferentemente do printf, colocando em um buffer o que se põe em seu protótipo. Por fim, faz-se o **write**, que escreve uma string em um arquivo, tendo em mãos o endereço de memória do buffer e o tamanho de bytes que se deseja escrever.

Assim, incluímos as bibliotecas:

- **stdio.h** – controle de entrada e saída
- **fcntl.h** – função open e suas flags
- **sys/types.h** – declarações de tipos utilizados pelo Minix

O bloco básico que fizemos, em cada uma das funções especificadas na seção anterior consiste em:

```
log = open("/var/log/memory.log", O_APPEND|O_CREAT|O_WRONLY);
/*criação/abertura do arquivo. log é uma variável que identificará o arquivo*/
tam = sprintf(buf, "<nome da syscall> <símbolo de inclusão ou retirada da memória (+ ou -)> processo %s +
        segmento 0x%lx - %d clicks\n",
        <processo>->mp_name, <processo>->mp_seg[D].mem_phys, <processo>->mp_seg[D].mem_len);
/*tam é o identificador de onde será guardada a string*/
write(log, buf, tam);
/*Ocrorrência da escrita, conforme especificado*/
```

Sendo assim, obtemos as seguintes alterações nos códigos fonte:

forkexit.c:

```
#include <sys/types.h>
#include <fcntl.h>
#include <stdio.h>

Função do_fork()
        int log, tam;
        char buf[64];

        ....

        log = open("/var/log/memory.log", O_APPEND|O_CREAT|O_WRONLY);
        tam = sprintf(buf, "fork - processo %s + segmento 0x%lx - %d clicks\n", rmc->mp_name,
                rmc->mp_seg[D].mem_phys, rmc->mp_seg[D].mem_len);
         write(log, buf, tam);

        close(log);
```

Observemos que é chamada a informação do processo filho, e que para a escrita, são utilizados os atributos especificados acima.

```
Função pm_exit()
        int log, tam;
        char buf[64];

        ....

        log = open("/var/log/memory.log", O_APPEND|O_CREAT|O_WRONLY);
        tam = sprintf(buf, "exit - processo %s + segmento 0x%lx - %d clicks\n", p_mp->mp_name,
                p_mp->mp_seg[D].mem_phys, p_mp->mp_seg[D].mem_len);
        write(log, buf, tam);


        close(log);
```

Agora, notemos que o processo chamado é o processo pai.

ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO

Avenida Professor Luciano Gualberto, travessa 3 nº 158 CEP 05508-900 São Paulo SP
Telefone: (11) 3091-5583 Fax (11) 3091-5294

Departamento de Engenharia de Computação e Sistemas Digitais

exec.c:

```
#include <sys/types.h>
#include <fcntl.h>
#include <stdio.h>

Função do_exec()
        int log, tam;
        char buf[64];

        ....

        log = open("/var/log/meexecfork - processo %s + segmento 0x%lx - %d clicks\n", rmp->mp_name,
                rmp->mp_seg[D].mem_phys, rmp->mp_seg[D].mem_len);
        write(log, buf, tam);

        close(log);
```

Por fim, novamente se chama o processo pai.

Com as alterações feitas, basta executar make install em /usr/src.

Apresentação de resultados

Por fim, os resultados obtidos foram bem satisfatórios, com a correta criação de um arquivo de log como especificado, vide screenshot: