Relatório EP 3 - MAC0422

Artur Magalhães R. Santos, 10297734

Caio Fontes, 10692061

Resumo

Nesse EP implementamos o utilitário *memorymap*, que imprime todos os processos ativos e o começo e o final de seus segmentos de memória, além da memória livre.

Também implementamos a syscall `memalloc`, que muda a política global de alocamento de memória entre *first-fit* e *worst-fit*.

Desenvolvimento

memorymap

No desenvolvimento do memory map começamos tentando criar uma nova syscall para repassar as informações do pm para o programa de usuário, isso se mostrou muito complexo e, ao analizarmos o código do programa *top.c*, percebemos qu a chamada `getsysinfo()` já fazia o que precisavamos.

O utilitário final acabou sendo, na prática, uma versão simplificada do *top.c*, já que quase toda a funcionalidade foi adaptada desse código original.

O código final está localizado em */usr/local/src* junto com um Makefile, o binário está em */usr/local/bin* e o script `compileMemMap.sh` compila o código e já move o executável para a pasta correta.

Worst-fit

A política foi implementada no arquivo */usr/src/servers/pm/alloc.c*, nele fizemos uma refatoração da função `alloc_mem()`, ela virou um wrapper que checa a variável global `ALLOC_POL` e chama a função que implementa a politica correta (`alloc_mem_worst_fit()` ou `alloc_mem_first_fit()`).

A função `alloc_mem_first_fit()` contém a implementação original para alocação de memória do MINIX.

A função `alloc_mem_worst_fit()` contém a nossa implementação para alocação de memória segundo a política *worst-fit*, que sempre utiliza um pedaço do maior espaço contíguo disponível na memória.

A variável global `ALLOC_POL` foi declarada no arquivo */usr/src/servers/pm/glo.h* e o arquivo */usr/src/servers/pm/alloc.h* foi criado para definição de algumas macros.

memalloc()

Ela foi implementada como uma chamada de sistema associada ao PM, seu código está em */usr/src/servers/pm/misc.c* e a função disponível para os usuários está definida em */usr/src/lib/posix/_memalloc.c* .

Ela faz a checagem de que o usuário que a chamou é o *root* e altera o valor de `ALLOC_POL`, alterando a política para todo o sistema.

Testes

O teste utilizado foi rodar o script a seguir e analizar os resultados obtidos:

```
#!/bin/sh

./make_holes &
memorymap > holes1.txt
./change_policy 1
./tester &
memorymap > holes_wf.txt
./make_holes &
memorymap > holes2.txt
./change_policy 0
./tester &
memorymap > holes_ff.txt
```

Código fonte de *change_policy*:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main(int argc, char** argv){
    int type = 0;

    if(argc > 1) type = atoi(argv[1]);

    return memalloc(type);
}
```

Código fonte de *make_holes*:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>


void make_child(){
    void* p;
    int child_pid = fork();
    if (child_pid < 0) {
        perror("Forking child has failed");
        exit(1);
    }
```

```
    if(child_pid) {
        p = malloc(2000);
        sleep(10);
        exit(0);
    }

    return;
}

int main(int argc, char** argv){

    int n_holes = 10,i;
    if(argc > 1) n_holes = atoi(argv[1]);

    for (i = 0; i< n_holes; i++)
        make_child();
    return 0;
}
```

Código fonte de *tester*:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main(int argc, char** argv){
    malloc(100000);
    sleep(40);
    return 0;
}
```

Pudemos observar que a politica estava sendo respeitada.

O Makefile deixado no diretório home compila os testes.