Relatório - EP Fase 4
Laboratório de Programação 2

23 de setembro de 2014

1 Integrantes

- Victor Sanches Portella - Nº USP: 7991152
- Vinícius Jorge Vendramini - Nº USP: XXXXXXX

2 Parte 1

Para a primeira parte do EP criamos uma função **dmp_ep** no arquivo *dmp_kernel.c*, onde ficava inicialmente a implementação da função chamada pelo **F4**. O funcionamento da função é simples e similar às outras funções de dump. São criados um vetor de processos e alguns contadores (estáticos, para possibilitar a paginação). Esse vetor é então ordenado pela prioridade e serve de fonte de dados para as impressões seguintes. As informações são tiradas então tanto do próprio vetor (vindo da tabela proc) quanto de uma outra tabela (a mproc) e impressas de acordo.

3 Parte 2

Para fazer a parte 2, criamos uma System Call chamada (setpriority_ep(pid, pri)), onde **pid** e **pri** são inteiros representando respectivamente o pid do processo alvo e a nova prioridade desse processo.

Essa System Call cria uma mensagem, que é enviada para o Process Manager, que chama a função **do_setpriority_ep()**. Lá verificamos se o processo chamador é pai do processo alvo, além de testar se o PID passado é de fato válido e se a prioridade dada como argumento é prioridade de usuário. Caso passe nos dois testes, chamamos a Kernel Call **sys_nice**,

que enviará uma mensagem para o System Task. Lá será verificado se a prioridade passada é válida,

Caso não haja nenhum problema, a prioridade é mudada na tabela de processos do Kernel, e o processo é mudado de fila de prioridade.

Valores de retorno possíveis da system call:[1]

- **Retorno 0:** Tudo ocorreu bem.
- **Retorno -1:** Prioridade inválida.
- **Retorno 1:** PID inválido ou o processo alvo não pode ter a prioridade alterada.
- **Retorno 2:** Processo alvo não é filho do processo chamador.

É importante notar que a prioridade **pri** passada é um número entre -20 e 20, não representando a real fila para a qual o processo será re-alocado. Quanto mais alto for **pri**, maior será a prioridade do processo. Dado esse número, o **sys_nice** faz uma conta para ter uma equivalência com relação ao número da fila que o processo deveria ir. Aqui fizemos uma tabela mostrando e qual fila o processo será realocado para cada valor de **pri**.

4 Teste

Nossa imagem do MINIX tem senha no root. Para acessar, use:

- **Login:** root
- **Senha:** senha

No endereço **/usr/src/EP2-testes** temos os arquivos de teste para verificar o funcionamento de nossa chamada ao sistema.

[1] Não seguimos o padrão do enunciado pois tivemos problemas ao tentar fazer a função retornar valor negativo, além de que ficaria ambíguo em determinados casos.