EP4 – MAC422 - Sistemas Operacionais

Daniela Gonzalez Favero 10277443
Felipe Castro de Noronha 10737032

Nesse exercício programa, mexemos no Sistema de Arquivos (FS) do Minix para adicionar **arquivos temporários**, isto é, arquivos que são removidos automaticamente quando o processo que os criou termina. Os passos da implementação e decisões de projeto são apresentados nos tópicos a seguir.

Todas as seções modificadas em arquivos estão entre os sinalizadores `/* EP4 ####...####*/`.

1. Um novo macro, para um novo i-node

Sabemos que no Minix, arquivos são implementados usando a estrutura de dados *i-node*. Logo, temos difentes rótulos de i-nodes sendo utilizados na estrutura interna do sistema, como, por exemplo, um rótulo específico para um diretório. Para o nosso arquivo temporário, vamos definir um novo tipo **I_TEMPORARY** e que será representado pela *flag* I_ **0030000**.

O arquivo em que este *define* foi criado é o `/usr/include/minix/const.h`.

2. Nova syscall open_tmp()

Para a criação dessa nova *syscall* definimos uma nova entrada na tabela de chamadas do FS. Logo, modificamos o arquivo `/usr/src/servers/fs/table.c`, adicionando o rótulo `do_open_tmp` na posição 70. Após isso, adicionamos o protótipo da função aplicada pela syscall no arquivo `/usr/src/servers/fs/proto.h`. A implementação dessa função foi realizada no arquivo `/usr/src/servers/fs/open.c`.

Para criar a syscall propriamente dita (uma função *bonitinha* que fica à mostra para o usuário) fizemos o seguinte: criamos um *define* que mapeia `OPEN_TMP` para o valor 70 (posição no vetor de funções do FS) nos arquivos `/usr/src/include/minix/callnr.h` e `/usr/include/minix/callnr.h`. Em seguida, implementamos o arquivo `/usr/src/lib/posix/_open_tmp.c`, que é responsável por receber a syscall e efetuá-la no FS, passando os argumentos recebidos, em uma mensagem. Após isso, bastou recompilar os servidores e bibliotecas e as chamadas estavam prontas.