Oitava aula de FSO

José A. Cardoso e Cunha
DI-FCT/UNL

Este texto resume o conteúdo da aula teórica.

1 Objectivo

O objectivo da aula foi a continuação do estudo do sistema de ficheiros e directorias, através do exemplo do sistema Unix.

2 Directorias

Existem dois níveis nos SF Unix: o nível simbólico, no qual se referem os ficheiros pelos seus nomes absolutos (*pathname*); o nível plano (*flat*), no qual se referem os ficheiros pelos seus índices i internos *i-node*.

É através do i-nome que o SO distingue os três tipos de ficheiro que reconhece: ficheiros normais em disco, directorias ou ficheiros especiais. Estes últimos correspondem aos dispositivos periféricos, os quais têm também nomes simbólicos (e.g. '/dev/tty1') e permitem as operações de *open, read, write*, sob o controlo de permissões, e com as devidas adaptações à sua natureza física.

No caso de um ficheiro normal ou de uma directoria, o i-node contém os endereços das estruturas de apontadores que permitem aceder aos blocos físicos que representam o ficheiro ou a directoria em disco. No caso de um ficheiro especial, o i-node contém os endereços das rotinas de controlo dos correspondentes dispositivos hardware. Estas rotinas, habitualmente designadas por *device drivers*, são invocadas na sequência das chamadas ao SO read e write, para as correspondentes operações serem efectuadas.

3 Sistema de ficheiros em disco

A figura 1 ilustram-se os elementos principais que representam a informação de um sistema de ficheiros Unix em disco.

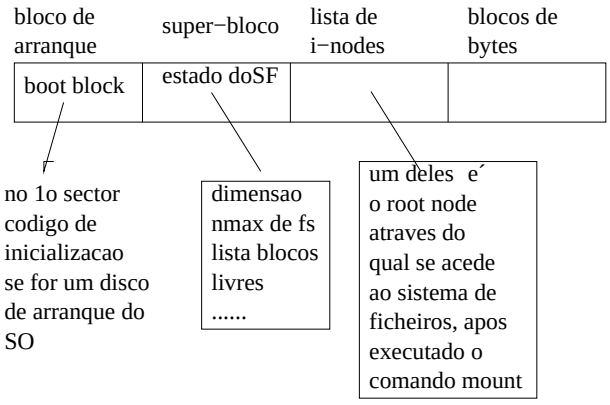

Figura 1: Um sistema de ficheiros Unix em disco

4 Abertura de um ficheiro: acções internas

Ao abrir um ficheiro, e.g. *open('/usr/d1/f',...)*, o SO localiza o seu i-node através da estrutura de directorias, a qual está inicialmente em disco. Possivelmente o i-node do nó raiz do sistema de ficheiros já está em memória, na sequência de anterior operação de montagem do sistema de ficheiros (veja o manual do Unix: operação mount). Na sequência da pesquisa do nome do ficheiro, o SO faz uma série de operações de leitura de disco, o que torna a operação *open* demorada, com tempos da escala de dezenas de milisegundos. A figura 2 ilustra os passos dessa pesquisa, para o exemplo dado acima.

Uma vez localizada a raiz do sistema de ficheiros, localiza-se o primeiro componente do nome '/usr' e o seu i-node (6). Um acesso a disco permite ler esse i-node para memória e ali encontrar o número do bloco (132) em disco onde está o conteúdo da directoria '/usr'. Uma vez trazido esse bloco para memória, pode ali localizar-se o segundo componente do nome ('d1') e o seu i-node (25). Um novo acesso a disco permite trazer para memória o bloco (406) da directoria d1. Ness bloco encontra-se o nome do ficheiro 'f1' e o seu i-node 45, pode ser localizado em disco e trazido para memória. Repare que o i-node, agora carregado em memória, contém, no caso de ficheiros normais em disco (ou de directorias) os endereços das estruturas cos apontadores dos blocos do ficheiro em disco, mas estes blocos só irão ser lidos para memória, à medida que o programa for pedindo operações de read ou de write.

Isto justifica que a operação *open* devolva um número de canal virtual, aberto para o ficheiro, e que dá acesso a estruturas de dados internas do SO, que são afectadas, no seguimento das acções acima descritas, na abertura de

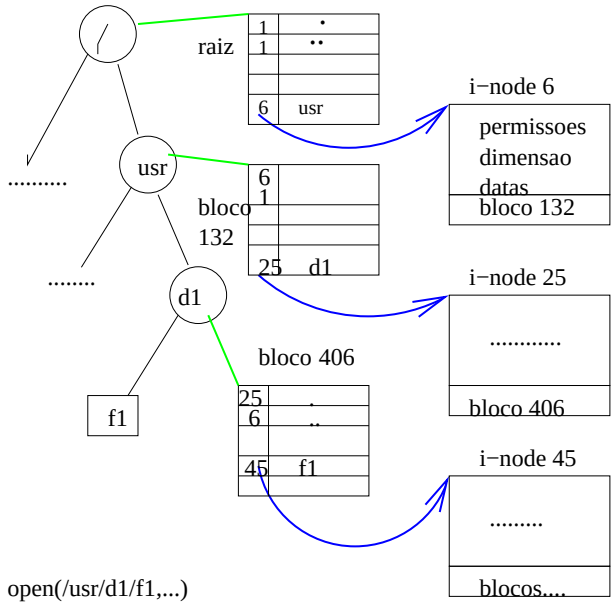

Figura 2: Abertura de um ficheiro Unix

um ficheiro. As principais estruturas são ilustradas na figura 3.

A tabela de i-nodes mantém em memória as cópias dos i-nodes que foram sendo lidos de disco, à medida que as operações *open* são invocadas. O i-node em memória de cada ficheiro dá acesso aos buffers de blocos do ficheiro, mantidos em memória pelo SO, e também dá acesso aos blocos do ficheiro em disco, no caso dos ficheiros normais e das directorias. Periodicamente a informação dos i-nodes em memória é escrita em disco.

A tabela global de ficheiros abertos tem uma entrada por cada canal aberto para um ficheiro. Essa entrada aponta para o i-node do ficheiro em memória, mas contém informação sobre o valor corrente do cursor do canal aberto para o ficheiro, bem como informação sobre o modo em que o ficheiro foi aberto: só leitura, só escrita ou leitura/escrita.

Há uma tabela de canais local a cada processo, cujas entradas são designadas por *file descriptors*. Essas entradas apontam a entrada correspondente ao canal, na tabela global de ficheiros do SO. O número da entrada do canal na tabela de canais corresponde ao valor devolvido pelas operações *create, open*.

Uma vez aberto um canal para um ficheiro e efectuado todo o complexo processamento da operação *open*, o ficheiro passa a ser acedido através do

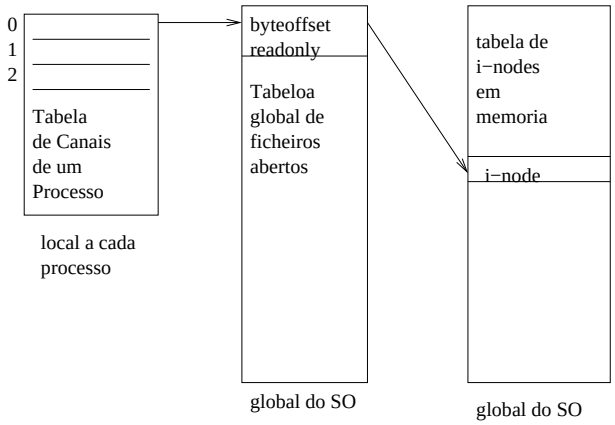

Figura 3: Tabelas de canais, de ficheiros abertos e de i-nodes

número de canal, o que tem a grande vantagem de o SO poder aceder eficientemente as estruturas do ficheiro, sem ter de voltar a pesquisar o seu nome nas directorias.

A figura 4 repete estas estruturas com maior detalhe.

Exemplo:

Considere as seguintes operações no processo P1:

open('f',O_RDONLY); open('g',O_WRONLY); open('f',O_RDWR);

e no processo P2:

open('f',O_RDONLY); open('h',O_RDONLY);

Isto origina a seguinte configuração de tabelas, ilustrada na figura 5.

Vê-se que os canais devolvidos no processo P1 foram, respectivamente, 3, 4 e 5, por se admitir que eram estas as primeiras três entradas livres da sua tabela de canais. No caso de P2, estes canais foram o 3 e o 4. Como se vê pela figura, não há relação alguma entre estes números, mesmo sendo iguais em processos diferentes, pois o seu significado é sempre relativo a uma entrada da tabela de canais local a cada processo.

Observe que, por cada canal aberto, há uma entrada na tabela de ficheiros abertos do SO. Nestas entradas há um contador, na figura com o valor 1 em todas as entradas. O contador é usado para que o SO saiba, quando um processo invoca a operação *close*, se não há mais referências de canais para essa entrada. Os dois únicos casos em que o contador desta entrada na tabela de ficheiros abertos é superior a 1, ocorrem quando um processo cria um processo filho (operação *fork*), ou quando um processo duplica explicitamente uma entrada da sua tabela de canais (ioeração *dup*).

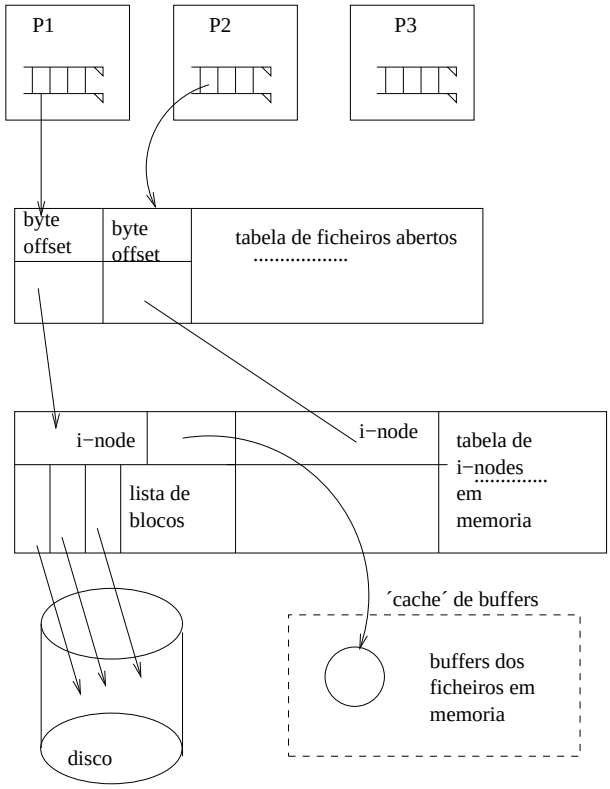

Figura 4: Tabelas de canais, de ficheiros abertos e de i-nodes

Note que há apenas, por cada ficheiro, uma entrada na tabela de i-nodes, pelo que aqui, no caso da entrada do ficheiro 'f', há um contador com o valor 3, pois temos três canais abertos para esse ficheiro. Quando houver um close() de um desses canais, o contador fica a 2, e por isso o i-node não é libertado de memória. Só quando o contador vier a 0, neste exemplo, após fechados os três canais, é que o i-node é libertado de memória, após ravada a informação em disco.

Note que, apesar de libertado o i-node em memória, o ficheiro continua a existir em disco. Novos canais poderão ser abertos para ele.

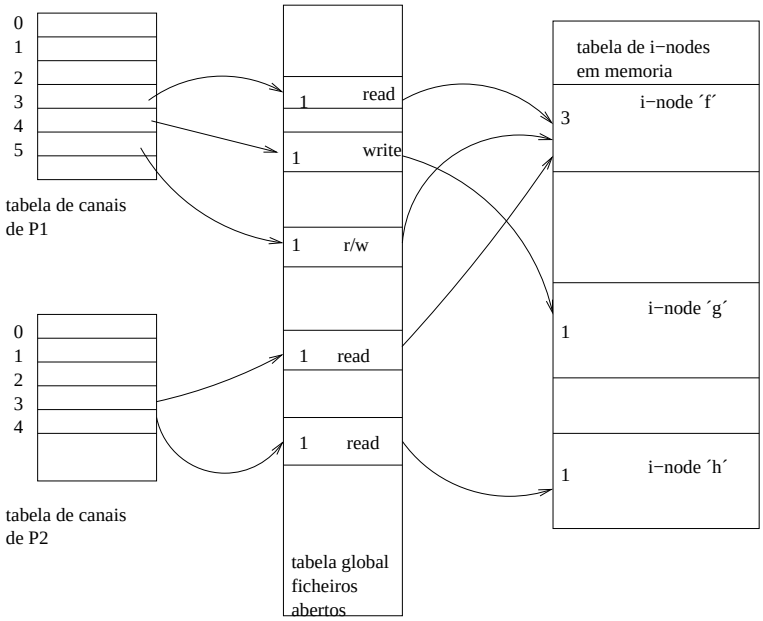

Figura 5: Tabelas de canais, de ficheiros abertos e de i-nodes