A complemento do trabalho prático no âmbito da unidade curricular de sistemas operativos, serve o presente relatório com o objetivo de melhor explanação de mecanismos e técnicas de programação usados.

Antes de mais será importante referir que a implementação foi primariamente adaptada do enunciado, principalmente da figura da segunda página.

Mecanismos de comunicação

Dado se pretender fazer a comunicação entre um processo servidor, em constante atividade, permitindo a receção de outros processos sob o formato de pedidos através de utilizadores, estes comunicam essencialmente através de fifos, dado a facilidade que estes proporcionam a que novos processos acedam aos mesmos canais de comunicação, estando estes representados por ficheiros com nome.
O server terá de ser sempre o primeiro processo a ser chamado e com ele cria-se a conta de administração e um fifo /tmp/secure_server aberto para leitura. No outro lado sempre que se chame um user, estes irão abrir o /tmp/secure_sever" para escrita, escrevendo o pedido a fazer ao servidor, através da estrutura tlv_request_t, vão também criar um fifo exclusivo, sendo identificado com o seu pid, aberto para leitura onde vão receber as respostas por parte do servidor, /tmp/secure_pid". Este fifo será aberto para escrita no servidor sempre que este processa um pedido nas threads auxiliares, no final do processamento o servidor escreverá aí a respetiva resposta sob formato tlv_reply_t.

Mecanismos de sincronização

Toda a implementação destes mecanismos foi baseada numa solução ao problema do produtor consumidor, não só por obrigação na própria descrição de trabalho mas também por se ajustar perfeitamente com o problema abordado.

Uma vez que, o server sempre que recebe informação por parte do user gera um pedido a ser tratado pelos offices, sendo estes threads auxiliares, isto significa que a main thread será o produtor gerando pedidos alocando-os numa fila, para que sejam tratados por ordem, que posteriormente serão consumidos nas threads ativas, que estarão em constante tratamento ou espera de dados a tratar.

Posto isto usam-se dois semáforos, um "full" inicializado com value igual a 0, para estar a par do número de pedidos que já existem no buffer, e outro "empty" inicializado com o value igual ao número de offices para estar a par das posições livres a serem ocupadas por produtos. Por último, é usado um mutex para garantir que não haja conflitos entre produtor e consumidores de modo a que ambos não possam aceder ao buffer ao mesmo tempo.

Assim, no produtor, main thread, inicialmente é decrementado o valor do semáforo empty, dado ser produzido um novo item, perdendo o buffer um espaço, isto faz-se com sem_wait(empty) para que caso empty esvazie bloqueie o processo ficando a espera de posições livres, após prosseguir o mutex bloqueia a thread garantido que nada é consumido no buffer ao mesmo tempo que outras threads, assim acede à secção crítica que aloca o

pedido na fila, por fim desbloqueia a mutex, e termina com sem_post(full) incrementando o valor de full.

Os vários consumidores nas threads auxiliares terão o mesmo procedimento que o produtor, no entanto invertendo a ordem dos semáforos, ou seja, inicialmente com sem_wait(full) decrementando o full, dado lhe ter sido retirado um request, e termina com sem_post(empty) libertando uma posição no buffer após ter sido tratado o pedido, o mutex é operado da mesma maneira que no produtor. No entanto neste caso a secção crítica será composta por um pop a fila e o tratamento de dados do pedido gerando um reply, através de um switch case consoante o tipo de pedido feito.

Server Shutdown

Quando o user envia um pedido de shutdown válido, o server recebe esse pedido e responde-lhe com o número de threads ativas, valor obtido através do semáforo empty. Após responder ao user, sai do ciclo dentro da thread e continua a execução normal, contudo, o ciclo na main thread vai ser alterado, uma vez que a condição foi alterada.
De seguida, na main thread, são mudadas as permissões de escrita do fifo /tmp/secure_srv, passando a ser apenas de leitura. Os pedidos que estavam na fila são concluídos e os que ainda estavam no buffer do fifo adicionados à fila. Quando o valor do semáforo empty for o inicial, significa que a fila está vazia, e que não temos mais pedidos a responder, então podemos destruir as threads e fechar, destruir e libertar a memória alocada para o programa e terminar a sua execução.