FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D COIMBRA

SO Relatório de Projeto

Carlos Soares

Pedro Lima

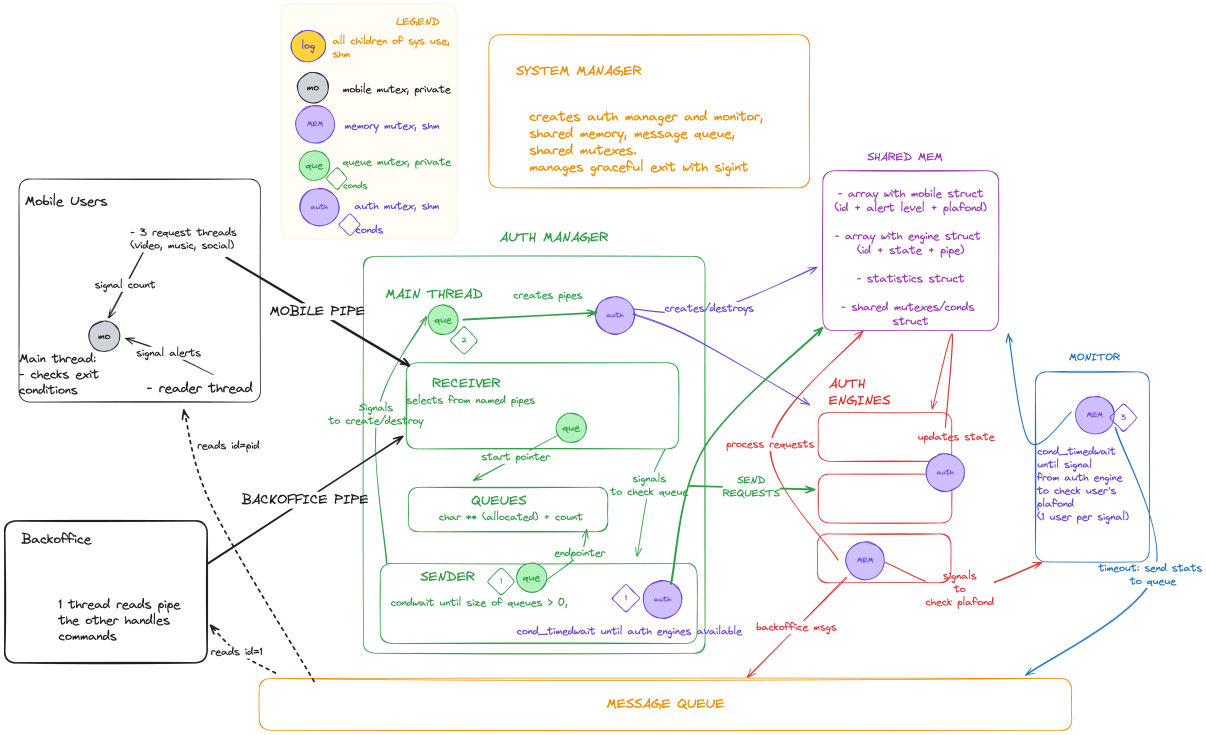
Bachelor's in Informatics Engineering
Faculty of Sciences and Technology of the University of Coimbra

May 14, 2024

1 Introdução

O presente relatório visa documentar brevemente as funcionalidades implementadas e as decisões tomadas no desenvolvimento do projeto final da Sistemas Operativos no ano letivo 23/24. O esforço de horas dos membros foi 40-60h Pedro Lima, 20-40h Carlos Soares.

2 Esquema



Como se pode identificar no esquema, foi utilizada uma combinação de mutexes privados e multi-processos para sincronizar o sistema.

Os pedidos são enviados pelas named pipes e selecionados pelo receiver, que os escreve na respetiva queue e manda um sinal ao sender para acordar. O sender está à espera (caso as queues estejam ambas vazias, caso contrário processa pedidos até o mesmo acontecer) no mesmo mutex do receiver, com uma variável de condição.

Depois de verificar o tamanho das queues, o sender pode mandar um sinal à main thread para criar/destruir o engine extra caso seja possível. Depois, liberta o mutex privado e tenta ganhar ownership do auth mutex, para confirmar um engine com estado ativo no sistema (precorrendo o array de "auth_struct" elementos na shared memory). Caso encontre manda-lhe a response pelo unnamed pipe aberto de antemão na shared memory pelo auth manager. Caso não encontre faz uma espera temporizada numa variável de condição também ela partilhada, até um auth engine assinalar mudança para estado ativo, ou dar timeout e descartar o pedido.

Do lado dos auth engines, estes usam o auth mutex apenas para atualizar o estado de volta a ativo (ele é atualizado para inativo pelo sender ao selecioná-lo). De resto, utiliza

outro mutex (mem) para aceder ao array de "mobile_struct" e às estatisticas contidas numa info_struct, ambas na shared memory, realizando as operações devidas após validar o request. Utiliza o mesmo mutex para enviar sinais ao monitor engine quando realiza requests de autorização bem sucedidos.

É de notar que todas as indicações para a shared memory são dadas através de ponteiros inicializados pelo sys manager. O monitor engine faz a espera do mesmo mutex com uma condição e timeout. Se o timeout ocorrer, envia as estatisticas para a message queue com identificador 1 (para ser acedido pelo backoffice user), caso contrário dá loop ao array de mobile users, encontrando o primeiro cujo consumo se encontra em condições de enviar um alerta para a message queue com o id desse user.

Já os mobile users são processos com 5 threads que enviam pedidos intercaladamente por uma named pipe para o sistema, escutam a message queue pelo identificador igual ao seu pid (que é obtido na primeira mensagem de registo desse user), com uma thread gestora que recebe sinais das outras threads para confirmar condições de saída do processo. O backoffice user usa duas threads, uma para escutar a message queue e imprimir mensagens recebidas, outra para escutar o stdin para permitir interação do utilizador.

O system manager está em pausa à espera dum sigint após inicializar a queue e a memória partilhada. Quando o recebe, envia um sigusr1 para ambos os processos filho e faz um waitpid aos 2, que por sua vez mudam uma end_flag privada a cada um para 1. O monitor engine como é single threaded e não tem recursos adicionais, simplesmente liberta o mutex e sai. O auth manager lida com a saída sincronizada escrevendo no pipe dos auth engines e fazendo um waitpid. Depois escreve no backoffice pipe uma mensagem de fecho para o receiver. O receiver por sua vez naturalmente segue o código, enviando o sinal ao sender que poderá ou não estar a espera na condição do mutex privado (queue vazia). Caso contrário, o sender chegará ao fim do loop até encontrar uma validação da end_flag (o mesmo acontece com o receiver). Para prevenir ações desnecessárias, validações são feitas a esta end_flag ao longo do código das várias threads. O auth manager faz join das threads, limpa as pipes e a memória alocada da queue, e o sys manager finalmente termina.

3 Correr o código

MO Makefile foi preparado bem como um script "mobile_users.sh" que recebe 1 argumento opcional {#users, default 5}. Todo o código foi testado em mac e ubuntu 20.04 arm64 com sucesso em todas as funcionalidades, tendo sido feito um esforço para emular as condições da VM amd64 disponibilizada. A maquina dei2 não permitia ao código permissões corretas para utilização de pipes e message queue.

4 Conclusão

Este projeto serviu para melhor compreender o funcionamento de processos no sistema operativo UNIX, e como a comunicação interprocessos pode ser feita, e o grupo sente-se realizado com o desafio e aprendizagem que este fomentou.