InterProcess Communication

Elaborado por:

Alexandre Santos nº52011
Vasco Colaço nº52290

Orientador:
PAUL ANDREW CROCKER

Capítulo 1

INTRODUÇÃO

Neste projeto foi desenvolvido um jogo do galo (tic-tac-toe) em linguagem C, com suporte para dois jogadores, utilizando comunicação entre processos através de named pipes (FIFOs). O sistema é composto por três módulos principais: um menu principal e dois executáveis correspondentes a cada jogador (jogador1 e jogador2).

O programa inicia com a execução do menu, que permite ao utilizador selecionar o jogador (1 ou 2), iniciar uma partida, reiniciar a pontuação ou encerrar o jogo. A comunicação entre o menu e os jogadores é realizadaatravés dos pipes nomeados `pipe_menu_jogadores` e `pipe_jogadores_menu`, pelos quais são enviadas as instruções e os resultados das partidas.

Cada jogador é executado como um processo independente, criado pelo menu através de chamadas ao fork() e execl(), e comunicam entre si utilizando dois pipes adicionais (pipe1to2 e pipe2to1). Durante o jogo, os jogadores trocam informações sobre o estado do tabuleiro e notificações sobre vitória ou empate, assegurando a sincronização e alternância correta de jogadas.

O projeto também inclui bibliotecas auxiliares responsáveis pela inicialização e visualização do tabuleiro, validação de jogadas, e verificação das condições de vitória ou empate.

Capítulo 2

IMPLEMENTAÇÃO

2.1 Arquitetura e Comunicação entre Processos

O sistema consiste em três programas principais: `menu`, `jogador1` e `jogador2`. O processo `menu` é responsável por inicializar o jogo, controlar o número de vitórias e lançar os processos dos jogadores. A comunicação entre o menu e os jogadores é realizada com os pipes:

- `pipe_menu_jogadores`: do menu para os jogadores (envio de indicação de quem inicia)
- `pipe_jogadores_menu`: dos jogadores para o menu (envio do resultado)
- `pipe1to2` e `pipe2to1`: para comunicação direta entre jogador1 e jogador2

Criação dos Pipes

```
#define PIPE1 "pipe_menu_jogadores"
#define PIPE2 "pipe_jogadores_menu"

mkfifo(PIPE1, 0666);
mkfifo(PIPE2, 0666);
```

Comunicação do Menu com os Jogadores

Quando o menu inicia um jogo, ele escreve no `PIPE1` o número do jogador que escolheu começar. O jogador lê essa informação para saber quem é. Após o jogo, o jogador vencedor (ou em caso de empate) envia o resultado para o menu através do `PIPE2`.

```
// Menu escreve no PIPE1
int fd = open(PIPE1, O_WRONLY);
write(fd, "1", 1); // Jogador 1 inicia
close(fd);

// Jogador envia resultado ao menu
int fd = open(PIPE2, O_WRONLY);
write(fd, "1", 1); // Jogador 1 venceu
close(fd);
```

2.2 Comunicação entre Jogadores

Para a troca de jogadas entre os jogadores, foram criados dois pipes adicionais:

- `pipe1to2`: jogador1 envia dados para jogador2
- `pipe2to1`: jogador2 envia dados para jogador1

Cada jogada consiste no envio do estado atualizado do tabuleiro e de um sinal de vitória ou empate.

```
// Jogador 1 envia jogada
char vitoria = 'V';
write(pipe1to2, &vitoria, 1);
write(pipe1to2, tabuleiro, sizeof(tabuleiro));

// Jogador 2 recebe
read(pipe1to2, &buffer, 1);
read(pipe1to2, tabuleiro, sizeof(tabuleiro));
```

2.3 Verificação de Vitória e Empate

A cada jogada, o programa verifica se houve vitória ou empate usando funções auxiliares:

```
if (verificar_vitoria(Tabuleiro, peca)) {
    write(caminhoJ1, "V", 1); // Vit ria
    write(caminhoJ1, Tabuleiro, sizeof(Tabuleiro));
    write(pipe_jogadores_menu, "1", 1); // Jogador 1 venceu
} else if (verificar_empate(Tabuleiro)) {
    write(caminhoJ1, "E", 1); // Empate
    write(caminhoJ1, Tabuleiro, sizeof(Tabuleiro));
    write(pipe_jogadores_menu, "E", 1);
}
```

Estas funções garantem que os resultados sejam comunicados corretamente ao adversário e ao menu, mantendo os dados sincronizados.

2.4 Conclusão

Este projeto permitiu aplicar conceitos fundamentais de comunicação entre processos com `pipes` nomeados. A implementação mostrou como coordenar múltiplos processos, sincronizar ações e partilhar estados de jogo de forma eficiente. A estrutura modular (menu, jogador1 e jogador2) tornou a comunicação clara e permitiu expandir funcionalidades como reiniciar pontuação ou terminar o jogo.