UFC

Modelagem de uma máquina de estados

TI0090/TIP711 - 2024.1 - Tópicos

Student name: Luiz Fernandes Menezes Lopes - 475396

Sumário

1 Objetivo

Procura-se implementar uma máquina de estados em C que simule o funcionamento do motor de um portão, com fechamento e abertura e um estado de emegência, caso necessário. A maquina de estados é descrita da seguinte forma:

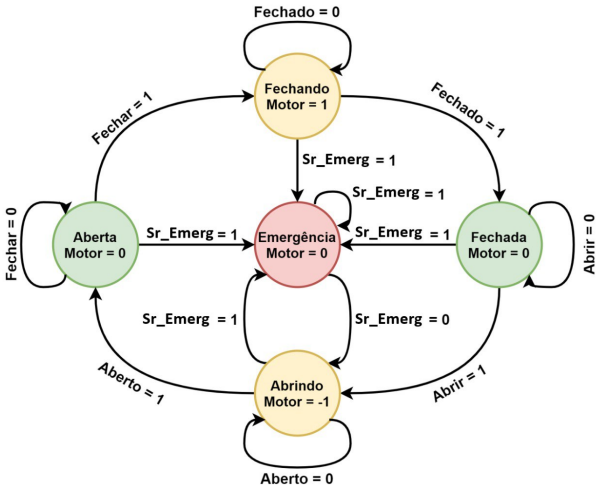

Figura 1: Esquema da máquina de estados

A figura representa uma máquina de estados para o controle de um motor, onde o motor pode estar em diferentes estados de operação. A transição entre esses estados é controlada por sinais de comando (Abrir, Fechar) e um sinal de emergência (`Sr_Emerg`). O sistema possui cinco estados principais descritos a seguir:

- **Estado 1 - Aberta (Motor = 0):** Neste estado, a porta controlada pelo motor está completamente aberta. A transição para o estado "Fechando"ocorre quando o comando `Fechar = 1` é ativado, enquanto a porta permanece aberta (`Fechar = 0`) caso nenhum comando seja acionado. Se houver uma emergência (`Sr_Emerg = 1`), o sistema transita para o estado de "Emergência".
- **Estado 2 - Fechada (Motor = 0):** Neste estado, a porta está completamente fechada. O sistema transita para o estado "Abrindo"quando o comando `Abrir = 1` é ativado. A porta permanecerá fechada (`Abrir = 0`) enquanto não houver comandos. Assim como nos outros estados, se ocorrer uma emergência (`Sr_Emerg = 1`), o sistema passa para o estado de "Emergência".
- **Estado 3 - Abrindo (Motor = -1):** Quando o comando de abertura (`Abrir = 1`) é ativado, o motor começa a abrir a porta e o sistema transita para o estado "Aberta"após o motor concluir a operação. Caso o comando de emergência seja ativado durante essa operação (`Sr_Emerg = 1`), o sistema transita para o estado de "Emergência".
- **Estado 4 - Fechando (Motor = 1):** Ao receber o comando `Fechar = 1`, o motor começa a fechar a porta e o sistema transita para o estado "Fechada"após a operação ser concluída. Se houver uma emergência enquanto o motor está fechando (`Sr_Emerg = 1`), a transição para o estado de "Emergência"é imediata.

- **Estado 5 - Emergência (Motor = 0):** Este estado é ativado sempre que o sinal de emergência (`Sr_Emerg = 1`) for detectado em qualquer outro estado. Nesse estado, o motor não realiza nenhuma operação de abertura ou fechamento. Para sair deste estado, o sistema deve reconhecer que o sinal de emergência foi resolvido (`Sr_Emerg = 0`), retornando ao estado "abrindo", como descrito no esquema fornecido..

As transições entre os estados são descritas pelos seguintes sinais:

- `Fechar = 1`: Transita de "Aberta"para "Fechando".
- `Abrir = 1`: Transita de "Fechada"para "Abrindo".
- `Sr_Emerg = 1`: Transita qualquer estado para "Emergência".
- `Fechar = 0`: Mantém o estado em "Aberta".
- `Abrir = 0`: Mantém o estado em "Fechada".
- `Sr_Emerg = 0`: Permite que o sistema saia do estado de "Emergência".

Cada estado da máquina é associado a um valor do motor, sendo 0 quando o motor está parado (nos estados "Aberta", "Fechada"e "Emergência"), 1 para o motor fechando a porta, e -1 para o motor abrindo a porta.

2 Introdução

Este documento descreve um código em C que simula o funcionamento de um portão, implementando um sistema de estados com capacidade de responder a comandos do usuário e gerenciar uma situação de emergência. É importante destacar que o código utiliza bibliotecas do Linux e que o portão inicialmente está fechado agurdando comandos. Todos os comandos exigem o uso da tecla 'Enter' e os dois comandos 'A' e 'E' são dados em maiúsculo. Uma versão comentada do código estará disponível no ZIP. Um ponto importante da resposta do código é que ele inicia no estado "fechado". É importante resssaltar que pelo uso das threads, as vezes, o comando 'A' ou 'E' não é computado, então é só repetir o comando.

Todos os comando são dados com letras maiúsculas.

3 Estruturas e Bibliotecas Utilizadas

O código inclui as seguintes bibliotecas:

- `stdio.h` - Para funções de entrada e saída.
- `stdlib.h` - Para funções auxiliares como `rand()`.
- `unistd.h` - Para a função `sleep()`.
- `pthread.h` - Para manipulação de threads.

- `fcntl.h` - Para manipulação de arquivos.
- `string.h` - Para manipulação de strings.
- `sys/select.h` - Para a função de verificação de entrada.

4 Definição dos Estados

O código define os seguintes estados possíveis do portão:

- `FECHADA` - O portão está fechado.
- `ABRINDO` - O portão está em processo de abertura.
- `ABERTA` - O portão está completamente aberto.
- `FECHANDO` - O portão está em processo de fechamento.
- `EMERGENCIA` - O portão entrou em modo de emergência.

5 Variáveis Globais

O código utiliza as seguintes variáveis globais:

- `emergencia` - Indicador de se uma emergência foi acionada (0 para não, 1 para sim).
- `estado_anterior` - Armazena o estado anterior do portão antes de uma emergência.

6 Funções Principais

O código contém várias funções principais:

- `main()` - Função principal que simula o funcionamento do portão e gerencia os estados.
- `abrir_portao(Estado *estado)` - Altera o estado para `ABRINDO`.
- `fechar_portao(Estado *estado)` - Altera o estado para `FECHANDO`.
- `ciclo_emergencia(Estado *estado)` - Executa o ciclo de emergência.
- `esperar(int segundos, Estado *estado)` - Simula um atraso e verifica se ocorreu uma emergência durante esse tempo.
- `verificar_emergencia(void *arg)` - Executa em uma thread separada para monitorar a entrada do usuário e detectar emergências.
- `input_disponivel()` - Verifica se há entrada disponível no stdin.

7 Fluxo de Execução

A execução do programa segue o fluxo abaixo:

1. O portão inicia no estado `FECHADA`.
2. O programa entra em um loop contínuo onde verifica o estado atual do portão.
3. Dependendo do estado, diferentes mensagens são exibidas e o usuário pode interagir pressionando teclas específicas ('A' para abrir ou fechar, 'E' para emergência).
4. Quando o portão é aberto ou fechado, o programa simula a operação com um atraso de alguns segundos.
5. Se a emergência for acionada, o estado muda para `EMERGENCIA`, e o portão é interrompido, aguardando 5 segundos antes de retornar ao estado `ABERTA`.

8 Código

[Motor de Estados]

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>
#include <fcntl.h>
#include <string.h>
#include <sys/select.h>

typedef enum {
    FECHADA,
    ABRINDO,
    ABERTA,
    FECHANDO,
    EMERGENCIA
} Estado;

int emergencia = 0;
Estado estado_anterior;

void abrir_portao(Estado *estado);
void fechar_portao(Estado *estado);
void ciclo_emergencia(Estado *estado);
void verificar_emergencia(void *arg);
void esperar(int segundos, Estado *estado);
int input_disponivel();
const char* nome_estado(Estado estado);

int main() {
    Estado estado = FECHADA;
    estado_anterior = estado;
```

```
    pthread_t thread_emergencia;
    pthread_create(&thread_emergencia, NULL, (void *) verificar_emergencia, NULL);

    while (1) {
        switch (estado) {
            case FECHADA:
                if (estado != estado_anterior) {
                    printf("portao est %s.\n", nome_estado(estado));
                    printf("Pressione 'A' para abrir.\n");
                    estado_anterior = estado;
                }
                if (input_disponivel()) {
                    char botao = getchar();
                    if (botao == 'A') {
                        abrir_portao(&estado);
                    }
                }
                break;

            case ABRINDO:
                if (estado != estado_anterior) {
                    printf("portao est %s...\n", nome_estado(estado));
                    estado_anterior = estado;
                }
                esperar(2, &estado);
                if (emergencia == 0) {
                    estado = ABERTA;
                    printf("portao est completamente %s.\n", nome_estado(estado)
                        );
                }
                break;

            case ABERTA:
                if (estado != estado_anterior) {
                    printf("portao est %s.\n", nome_estado(estado));
                    printf("Pressione 'A' para fechar.\n");
                    estado_anterior = estado;
                }
                if (input_disponivel()) {
                    char botao = getchar();
                    if (botao == 'A') {
                        fechar_portao(&estado);
                    }
                }
                break;

            case FECHANDO:
                if (estado != estado_anterior) {
                    printf("portao est %s...\n", nome_estado(estado));
                    estado_anterior = estado;
                }
                while (estado == FECHANDO) {
                    esperar(1, &estado);
                    if (emergencia == 0) {
```

```
                        printf("Continuando a fechar...\n");
                    } else {
                        break;
                    }

                    if (rand() % 3 == 0) {
                        estado = FECHADA;
                        printf("portao est completamente %s.\n", nome_estado(
                            estado));
                    }
                }
                break;

            case EMERGENCIA:
                if (estado_anterior != EMERGENCIA) {
                    printf("portao em %s! Parando o portao.\n", nome_estado(estado
                        ));
                }
                ciclo_emergencia(&estado);
                if (estado == EMERGENCIA) {
                    printf("Emerg ncia resolvida. Iniciando abertura do port o.\
                        n");
                    estado = ABRINDO;
                    emergencia = 0;
                }
                break;

            default:
                printf("Estado desconhecido!\n");
        }
    }

    return 0;
}

void abrir_portao(Estado *estado) {
    printf("Iniciando abertura do portao...\n");
    *estado = ABRINDO;
}

void fechar_portao(Estado *estado) {
    printf("Iniciando fechamento do portao...\n");
    *estado = FECHANDO;
}

void ciclo_emergencia(Estado *estado) {
    printf("Por favor, resolva a emerg ncia...\n");
    sleep(5);
    printf("Emerg ncia resolvida.\n");
}

void esperar(int segundos, Estado *estado) {
    for (int i = 0; i < segundos; i++) {
        sleep(1);
```

```
        if (emergencia == 1) {
            if (*estado != EMERGENCIA) {
                estado_anterior = *estado;
            }
            *estado = EMERGENCIA;
            break;
        }
    }
}

void verificar_emergencia(void *arg) {
    while (1) {
        if (input_disponivel()) {
            char botao = getchar();
            if (botao == 'E') {
                emergencia = 1;
            }
        }
    }
}

int input_disponivel() {
    fd_set readfds;
    struct timeval tv;

    FD_ZERO(&readfds);
    FD_SET(STDIN_FILENO, &readfds);
    tv.tv_sec = 0;
    tv.tv_usec = 0;

    return select(1, &readfds, NULL, NULL, &tv) > 0;
}

const char* nome_estado(Estado estado) {
    switch (estado) {
        case FECHADA: return "FECHADA";
        case ABRINDO: return "ABRINDO";
        case ABERTA: return "ABERTA";
        case FECHANDO: return "FECHANDO";
        case EMERGENCIA: return "EMERGENCIA";
        default: return "DESCONHECIDO";
    }
}
```