UNIVERSIDADE DE PERNAMBUCO

SIMULACÃO DE REDE DE PETRI COM ENTRADA VARIÁVEL

PROFESSOR: DR. RUBEN CARLO BENANTE

NOMES: JOÃO PEDRO PACHECO RODRIGUES ALMEIDA

PAULO VÍTOR ALVES PATRIOTA

RECIFE, 2016.

O projeto, realizado no Linux com orientação do professor, visa simular e representar uma determinada rede de petri. O algoritmo pode ser dividido em três partes: entrada, processamento e representação. Como pode ser visto na função main() no programa.

```
int main(void)
{
    transicoes *cabeca_transicoes=NULL;
    estados *cabeca_estados=NULL;
    tadt *cabeca_threads=NULL;
    gerar_entrada(&cabeca_estados, &cabeca_transicoes);
    printf("\nANTES DA SIMULACAO");
    debug(cabeca_estados, cabeca_transicoes);
    criar_threads(&cabeca_threads, cabeca_transicoes, cabeca_estados);
    espera_threads(cabeca_threads);
    gerar_imagem(cabeca_transicoes);
    printf("\nDEPOIS DA SIMULACAO");
    debug(cabeca_estados, cabeca_transicoes);
    return 0;
}
```

debug() é uma função para impressão dos dados na tela apenas para certificação de que o programa está funcionando. Esta função pode ser comentada ou retirada assim como as funções printf() da função main.

Para entendimento do algoritmo o leitor deve ter um conhecimento básico de uma rede petri. Esta contém apenas três elementos: estados ou lugares, transições e arcos. Os estados ou lugares são as posições onde podem ser acumulados tokens e representam algo na determinada aplicação que a rede seja baseada. Uma máquina, exemplo. Tokens são os indicadores de que o lugar ou estado está pronto para passar o suprimento à diante, representa um ciclo de máquina, por exemplo. As transições são responsáveis por transferir os tokens de um estado para o outro, são as ações de um operário no chão de fábrica ao transferir peças de um ponto para o outro. Esta ação é atômica, ou seja, a utilização dos tokens de custo de produção e dos tokens de saída devem ocorrer no mesmo momento. Os arcos representam o custo que esta transferência tem e quanto que ela vai produzir, cem parafusos para montar um equipamento.

A entrada é realizada com um arquivo no formato .txt onde a rede de petri é previamente escrita com um formato predeterminado da seguinte forma:

```
---------- INICIO DO ARQUIVO 1 ----------
4 # total de lugares
2 # total de 2 transições
2 # tem 2 lugares com tokens > 0 (2 linhas do grupo 1)
3 # tem 3 arcos de lugar->transição (3 linhas do grupo 2)
3 # tem 3 arcos de transição->lugar (3 linhas do grupo 3)
0 6 # grupo 1, linha 1, lugar 0 tem 6 tokens
2 1 # grupo 1, linha 2, lugar 2 tem 1 token
0 2 0 # grupo 2, linha 1, arco de lugar 0, gasta 2 tokens, vai para transição 0
```

1 1 1 # grupo 2, linha 2, arco de lugar 1, gasta 1 token, vai para transição 1
2 1 1 # grupo 2, linha 3, arco do lugar 2, gasta 1 token, vai para transição 1
0 1 1 # grupo 3, linha 1, arco da transição 0, gasta 1 token, vai para o lugar 1
1 1 2 # grupo 3, linha 2, arco da transição 1, gasta 1 token, vai para o lugar 2
1 1 3 # grupo 3, linha 3, arco da transição 1, gasta 1 token, vai para o lugar 3
---------- FIM DO ARQUIVO 1 ----------

O arquivo deve conter apenas números, tudo que foi escrito após o jogo da velha são comentários produzidos pelo professor para elucidar a turma. Esta entrada deve ser transferida para o programa onde será processada e simulada.

A simulação consiste em utilizar da entrada, já transferida para linguagem de máquina, processando da forma mais otimizada possível para reduzir o tempo que o processor será utilizado. Todas as transições devem existir simultaneamente e independentes umas das outras, foi a única regra estabelecida para o projeto. Portanto, cada uma delas deve ser representada por uma thread diferente de forma que qualquer uma pode ser ativada a qualquer momento como mostrado na função criar_threads(). Isto é feito com o seguinte código:

```
while(pt!=NULL)
{
        if(DEBUG)
            printf("Thread da transicao %d sendo criada.\n", pt->tr->ntr);
        pthread_create(&(pt->nth), NULL, roda_thread, (void *)pt);
        pt=pt->prox;
}
```

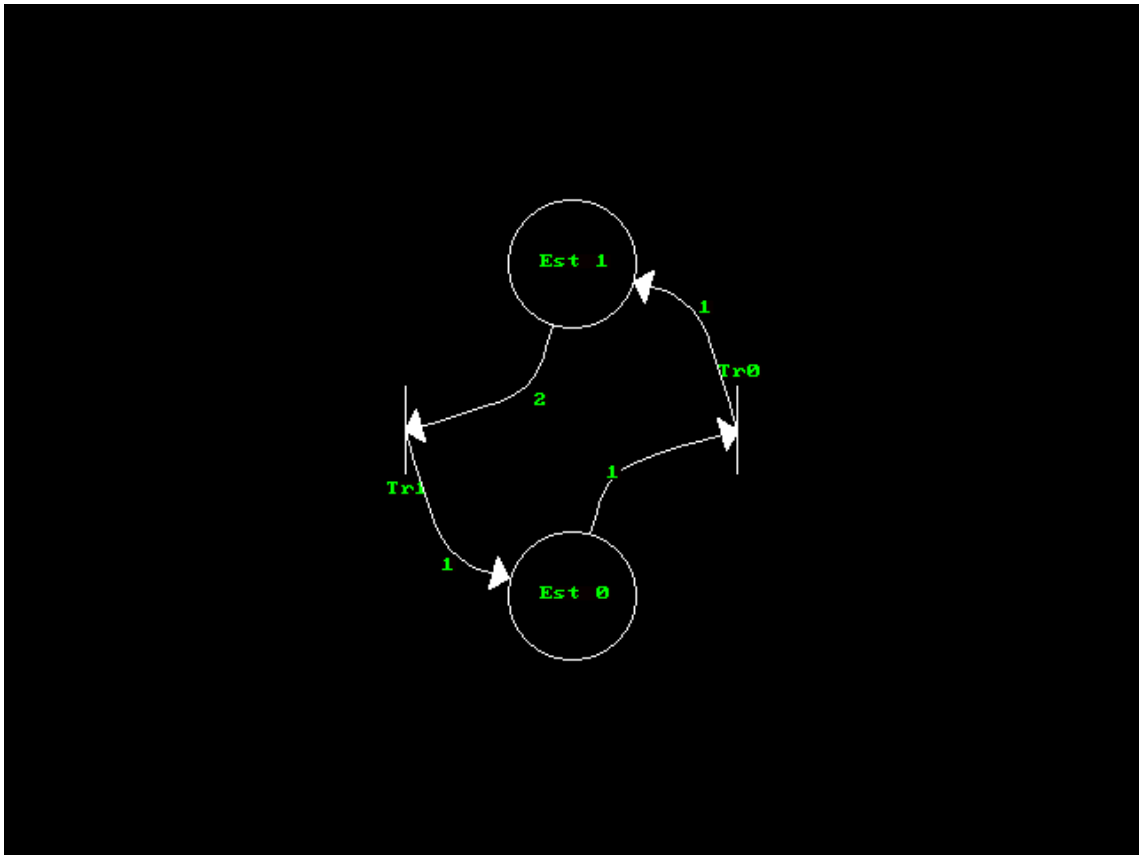
Onde *pt é um ponteiro para a estrutura que armazena as informações necessárias para a criação da thread e desenvolvimento da thread.

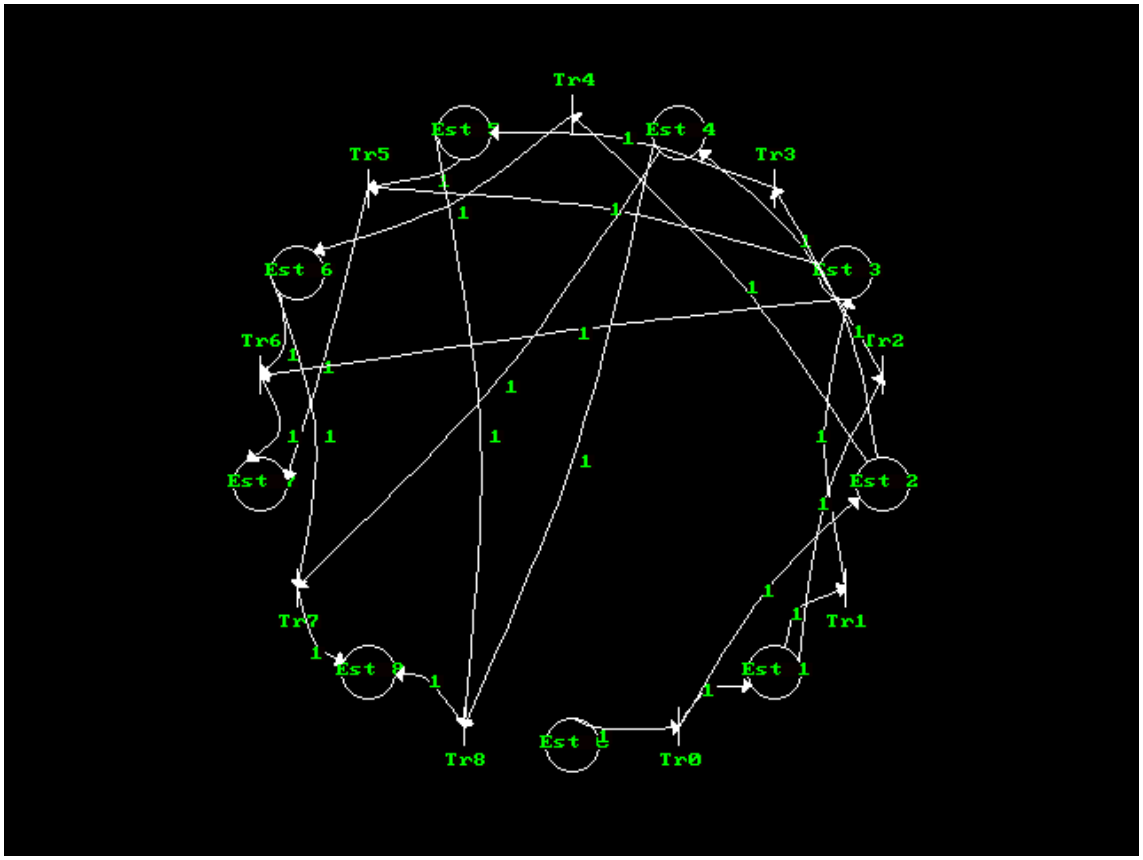
```
typedef struct stadt /*Estrutura para auxiliar na criacao das threads.*/
{
    pthread_t nth;
    struct stransicoes *tr; /*Transicao que a thread esta rodando.*/
    struct sestados *std; /*Ira armazenar a cabeca dos estados.*/
    struct stadt *prox;
}tadt;
```

Após a simulação a representação da rede deve ser realizada com o allegro, biblioteca livre de código fonte aberto. Muito destinada, também, paraprodução de jogos. A saída deve ser um arquivo do tipo .bmp.

Segue no final dois exemplos de rede de petri desenhadas utilizando a biblioteca mencionada.

Como considerações finais devo recomendar a criação de um programa auxiliar para criar as entradas do simulador pois fazer o arquivo de entrada pode ser tornar cansativo a depender do tamanho da rede a ser simulada.


Est 1
Tr0
Tr1
Est 0


Tr4
Est 7
Est 4
Tr5
Tr3
Est 6
Est 3
Tr6
Tr2
Est 5
Est 2
Tr7
Tr1
Est 0
Est 1
Est 8
Tr8
Tr0