INF1316 - SISTEMAS OPERACIONAIS - 2022.2 - 3WA
Tarefa 2 - Threads, tempo sequencial e tempo paralelo, comparação com processos

Nome: Eric Leão Matrícula: 2110694
Nome: Pedro Machado Peçanha Matrícula: 2110535

Conseguimos fazer o trabalho sem maiores dificuldades, e acreditamos que nada esteja faltando. Para rodar os processos no fundo, utilizamos a função execl.
Separamos 3 ponteiros, sendo que cada um tinha uma função dentro do programa. Um apontava para uma fila de criação (c_queue), que era, basicamente, uma fila que possuía todos os elementos que deveriam ser criados pelo programa, ordenados de forma que o programa que tivesse o menor tempo de criação ficasse na primeira posição da fila (ordenação via qsort). O segundo aponta para os processos que já foram criados, em ordem de prioridade, do maior para o menor, atuando como uma fila de pronto (r_queue). Como podemos ter processos com a mesma prioridade, sempre que adicionamos um novo processo, colocamos ele atrás de todos os processos com prioridade maior ou igual à dele e à frente de todos os que têm prioridade menor. O último aponta para o processo em execução no momento (aux).
O programa é regido por um loop principal, que só para quando não temos mais um processo corrente, uma fila de pronto ou uma fila de criação. A cada rodada do loop, checamos se existe algum processo que deve ser criado naquele momento na fila de criação. Se sim, é criado e colocado na fila de pronto, na posição adequada de acordo com sua prioridade. Depois, checamos se o processo que está na fila de pronto primeiro é mais prioritário do que o que está em execução. Se sim, o processo em execução volta para a fila de pronto. Após isso, checamos se o que está em execução tem a mesma prioridade do atual da fila de pronto. Se sim, executamos em round-robin, substituindo o processo corrente pelo da fila de pronto alternadamente. Ao final, diminuímos 1 unidade do tempo total de execução do processo (burst) e, se esse valor chegar a zero, matamos o processo. Ao final, também, sempre salvamos o tempo atual, o processo que está sendo executado, a fila de criação e a fila de pronto em um arquivo chamado “resposta.txt”.

exemplo1:

exec p1, prioridade=3, inicio_tempo_execucao=0, tempo_total_ execucao =6
exec p2, prioridade=3, inicio_tempo_execucao=0, tempo_total_ execucao =4
exec p3, prioridade=2, inicio_tempo_execucao=4, tempo_total_ execucao =3

resultado1:

tempo = 0
executando = p1
fila de pronto = - p2
fila de criacao = - p3

tempo = 1
executando = p1
fila de pronto = - p2
fila de criacao = - p3

tempo = 2
executando = p1
fila de pronto = - p2
fila de criacao = - p3

tempo = 3
executando = p2
fila de pronto = - p1
fila de criacao = - p3

tempo = 4
executando = p3
fila de pronto = - p1 - p2
fila de criacao =

tempo = 5
executando = p3
fila de pronto = - p1 - p2
fila de criacao =

tempo = 6
executando = p3
fila de pronto = - p1 - p2
fila de criacao =

tempo = 7
executando = p1
fila de pronto = - p2
fila de criacao =

tempo = 8
executando = p1
fila de pronto = - p2
fila de criacao =

tempo = 9
executando = p1
fila de pronto = - p2
fila de criacao =

tempo = 10
executando = p2
fila de pronto =
fila de criacao =

tempo = 11
executando = p2

fila de pronto =
fila de criacao =

tempo = 12
executando = p2
fila de pronto =
fila de criacao =

Todos os processos foram executados com sucesso!

exemplo 2:

exec p1, prioridade=3, inicio_tempo_execucao=0, tempo_total_ execucao =8
exec p2, prioridade=2, inicio_tempo_execucao=2, tempo_total_ execucao =9
exec p3, prioridade=1, inicio_tempo_execucao=3, tempo_total_ execucao =11
exec p4, prioridade=2, inicio_tempo_execucao=4, tempo_total_ execucao =13
exec p5, prioridade=1, inicio_tempo_execucao=5, tempo_total_ execucao =14

resultado 2:

tempo = 0
executando = p1
fila de pronto =
fila de criacao = - p2 - p3 - p4 - p5

tempo = 1
executando = p1
fila de pronto =
fila de criacao = - p2 - p3 - p4 - p5

tempo = 2
executando = p2
fila de pronto = - p1
fila de criacao = - p3 - p4 - p5

tempo = 3
executando = p3
fila de pronto = - p2 - p1
fila de criacao = - p4 - p5

tempo = 4
executando = p3
fila de pronto = - p2 - p4 - p1
fila de criacao = - p5

tempo = 5
executando = p3
fila de pronto = - p5 - p2 - p4 - p1

fila de criacao =

tempo = 6
executando = p5
fila de pronto = - p3 - p2 - p4 - p1
fila de criacao =

tempo = 7
executando = p5
fila de pronto = - p3 - p2 - p4 - p1
fila de criacao =

tempo = 8
executando = p5
fila de pronto = - p3 - p2 - p4 - p1
fila de criacao =

tempo = 9
executando = p3
fila de pronto = - p5 - p2 - p4 - p1
fila de criacao =

tempo = 10
executando = p3
fila de pronto = - p5 - p2 - p4 - p1
fila de criacao =

tempo = 11
executando = p3
fila de pronto = - p5 - p2 - p4 - p1
fila de criacao =

tempo = 12
executando = p5
fila de pronto = - p3 - p2 - p4 - p1
fila de criacao =

tempo = 13
executando = p5
fila de pronto = - p3 - p2 - p4 - p1
fila de criacao =

tempo = 14
executando = p5
fila de pronto = - p3 - p2 - p4 - p1
fila de criacao =

tempo = 15

executando = p3
fila de pronto = - p5 - p2 - p4 - p1
fila de criacao =

tempo = 16
executando = p3
fila de pronto = - p5 - p2 - p4 - p1
fila de criacao =

tempo = 17
executando = p3
fila de pronto = - p5 - p2 - p4 - p1
fila de criacao =

tempo = 18
executando = p5
fila de pronto = - p3 - p2 - p4 - p1
fila de criacao =

tempo = 19
executando = p5
fila de pronto = - p3 - p2 - p4 - p1
fila de criacao =

tempo = 20
executando = p5
fila de pronto = - p3 - p2 - p4 - p1
fila de criacao =

tempo = 21
executando = p3
fila de pronto = - p5 - p2 - p4 - p1
fila de criacao =

tempo = 22
executando = p3
fila de pronto = - p5 - p2 - p4 - p1
fila de criacao =

tempo = 23
executando = p5
fila de pronto = - p2 - p4 - p1
fila de criacao =

tempo = 24
executando = p5
fila de pronto = - p2 - p4 - p1
fila de criacao =

tempo = 25
executando = p5
fila de pronto = - p2 - p4 - p1
fila de criacao =

tempo = 26
executando = p5
fila de pronto = - p2 - p4 - p1
fila de criacao =

tempo = 27
executando = p5
fila de pronto = - p2 - p4 - p1
fila de criacao =

tempo = 28
executando = p2
fila de pronto = - p4 - p1
fila de criacao =

tempo = 29
executando = p2
fila de pronto = - p4 - p1
fila de criacao =

tempo = 30
executando = p2
fila de pronto = - p4 - p1
fila de criacao =

tempo = 31
executando = p4
fila de pronto = - p2 - p1
fila de criacao =

tempo = 32
executando = p4
fila de pronto = - p2 - p1
fila de criacao =

tempo = 33
executando = p4
fila de pronto = - p2 - p1
fila de criacao =

tempo = 34
executando = p2

fila de pronto = - p4 - p1
fila de criacao =

tempo = 35
executando = p2
fila de pronto = - p4 - p1
fila de criacao =

tempo = 36
executando = p2
fila de pronto = - p4 - p1
fila de criacao =

tempo = 37
executando = p4
fila de pronto = - p2 - p1
fila de criacao =

tempo = 38
executando = p4
fila de pronto = - p2 - p1
fila de criacao =

tempo = 39
executando = p4
fila de pronto = - p2 - p1
fila de criacao =

tempo = 40
executando = p2
fila de pronto = - p4 - p1
fila de criacao =

tempo = 41
executando = p2
fila de pronto = - p4 - p1
fila de criacao =

tempo = 42
executando = p4
fila de pronto = - p1
fila de criacao =

tempo = 43
executando = p4
fila de pronto = - p1
fila de criacao =

tempo = 44
executando = p4
fila de pronto = - p1
fila de criacao =

tempo = 45
executando = p4
fila de pronto = - p1
fila de criacao =

tempo = 46
executando = p4
fila de pronto = - p1
fila de criacao =

tempo = 47
executando = p4
fila de pronto = - p1
fila de criacao =

tempo = 48
executando = p4
fila de pronto = - p1
fila de criacao =

tempo = 49
executando = p1
fila de pronto =
fila de criacao =

tempo = 50
executando = p1
fila de pronto =
fila de criacao =

tempo = 51
executando = p1
fila de pronto =
fila de criacao =

tempo = 52
executando = p1
fila de pronto =
fila de criacao =

tempo = 53
executando = p1
fila de pronto =

fila de criacao =

tempo = 54
executando = p1
fila de pronto =
fila de criacao =

Todos os processos foram executados com sucesso!

código:

```
#include <assert.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/shm.h>
#include <sys/stat.h>
#include <sys/wait.h>
#include <time.h>
#include <unistd.h>

#define BUFF 100
FILE *resp;

typedef struct process {
  pid_t pid;
  char name[100];
  int priority;
  int startTime;
  int totalTime;
  int burst;
} Process;

typedef struct _queue {
  Process *process_node;
  struct _queue *process_nxt;
} queue;

int cmpfunc(const void *a, const void *b) {
  Process *pa = (Process *)a;
  Process *pb = (Process *)b;
  return (pa->startTime - pb->startTime);
}

queue *ready();
queue *creating(Process *arr, int c);
Process create_process(Process process, char);
queue *node_remover(queue *r_queue);
```

```
queue *node_add(queue *r_queue, Process *p);
void schedule(queue *r_queue, queue *c_queue);

int main(int argc, char *argv[]) {
  int c = 0;
  FILE *arq;
  arq = fopen(argv[1], "r");
  resp = fopen("resposta.txt", "w");
  assert(arq);
  Process processArray[BUFF];
  while (fscanf(arq, "exec %[a-zA-Z0-9]", processArray[c].name) == 1) {
          assert(fscanf(arq, ", prioridade=%d", &processArray[c].priority) == 1);
          assert(fscanf(arq, ", inicio_tempo_execucao=%d",
                  &processArray[c].startTime) == 1);
          assert(fscanf(arq, ", tempo_total_ execucao =%d\n",
                  &processArray[c].totalTime) == 1);
          processArray[c].burst = processArray[c].totalTime;
          processArray[c].pid = -1;
          c++;
  }
  schedule(ready(), creating(processArray, c));
  fclose(arq);
  fclose(resp);
  return 0;
}

queue *ready() { return NULL; }
queue *creating(Process *arr, int c) {
  queue *raiz;
  raiz = (queue *)malloc(sizeof(queue));
  queue *aux = raiz;
  qsort(arr, c, sizeof(Process), cmpfunc);
  for (int i = 0; i < c; i++) {
          aux->process_node = &arr[i];
          if (i != c - 1) {
          queue *novo;
          novo = (queue *)malloc(sizeof(queue));
          aux->process_nxt = novo;
          novo->process_nxt = NULL;
          aux = novo;
          }
  }
  return raiz;
}

void schedule(queue *r_queue, queue *c_queue) {
  int time = 0;
  int id;
```

```
Process *aux = NULL;
int ts = 3;
while (r_queue || c_queue || aux) {
        while (c_queue && c_queue->process_node->startTime == time) {
        r_queue = node_add(r_queue, c_queue->process_node);
        c_queue = node_remover(c_queue);
        }
        if (r_queue && aux == NULL) {
        aux = r_queue->process_node;
        if (aux->pid == -1) {
        if ((id = fork()) == 0) {
        execl(aux->name, aux->name, NULL);
        }
        aux->pid = id;
        } else {
        kill(aux->pid, SIGCONT);
        }
        r_queue = node_remover(r_queue);
        ts = 3;
        } else if (r_queue && aux &&
        (aux->priority > r_queue->process_node->priority)) {
        r_queue = node_add(r_queue, aux);
        kill(aux->pid, SIGSTOP);
        aux = r_queue->process_node;
        if (aux->pid == -1) {
        if ((id = fork()) == 0) {
        execl(aux->name, aux->name, NULL);
        }
        aux->pid = id;
        } else {
        kill(aux->pid, SIGCONT);
        }
        r_queue = node_remover(r_queue);
        ts = 3;
        } else if (r_queue && aux &&
        (ts <= 0 &&
                (aux->priority == r_queue->process_node->priority))) {
        r_queue = node_add(r_queue, aux);
        kill(aux->pid, SIGSTOP);
        aux = r_queue->process_node;
        if (aux->pid == -1) {
        if ((id = fork()) == 0) {
        execl(aux->name, aux->name, NULL);
        }
        aux->pid = id;
        } else {
        kill(aux->pid, SIGCONT);
        }
```

```
        r_queue = node_remover(r_queue);
        ts = 3;
        }
        fprintf(resp, "tempo = %d\n", time);
        if (aux)
        fprintf(resp, "executando = %s\n", aux->name);
        else
        fprintf(resp, "nenhum programa executando\n");
        queue *auxq = r_queue;
        fprintf(resp, "fila de pronto = ");
        while (auxq != NULL) {
        fprintf(resp, " - %s", auxq->process_node->name);
        auxq = auxq->process_nxt;
        }
        fprintf(resp, "\n");
        queue *auxq1 = c_queue;
        fprintf(resp, "fila de criacao = ");
        while (auxq1 != NULL) {
        fprintf(resp, " - %s", auxq1->process_node->name);
        auxq1 = auxq1->process_nxt;
        }
        fprintf(resp, "\n\n");
        if (aux) {
        aux->burst--;
        if (aux->burst == 0) {
        kill(aux->pid, SIGKILL);
        aux = NULL;
        }
        }
        time++;
        ts--;
        sleep(1);
  }
  fprintf(resp, "Todos os processos foram executados com sucesso!");
}

queue *node_add(queue *r_queue, Process *p) {
  queue *novo;
  novo = (queue *)malloc(sizeof(queue));
  novo->process_node = p;
  queue *current = r_queue;
  if (r_queue == NULL || r_queue->process_node->priority > p->priority) {
        novo->process_node = p;
        novo->process_nxt = r_queue;
        return novo;
  }
  while (current->process_nxt != NULL &&
        current->process_nxt->process_node->priority <= p->priority) {
```

```
            current = current->process_nxt;
  }
  novo->process_nxt = current->process_nxt;
  current->process_nxt = novo;
  return r_queue;
}

queue *node_remover(queue *r_queue) {
  queue *ret = r_queue->process_nxt;
  free(r_queue);
  return ret;
}
```