Apresentação Sistemas Operacionais - Trabalho 1

Dimitri Barreto Medeiros
Victor Medeiros Martins

Estratégia Inicial

Criação de 3 processos:

- Processo 1 (Atendimento)
- Processo 2 (Analista)
- Processo 3 (Cliente)

Os três estão separados em arquivos distintos “.c”.

Usamos Makefile para compilar. (Flag -o3 para otimizações do compilador)

Executamos ./atendimento N P

Onde “N” é o número de clientes que serão criados, e “P” é a paciência, podendo ser cortada pela metade na execução para clientes de alta prioridade.

atendimento.c - Threads

Ao executar, cria-se um processo atendimento, onde:

- Cria 1 thread service (ou Atendente)
- Cria 1 thread reception (ou Recepção)

Além disso, cria-se uma thread auxiliar a fim de parar todas as threads(stopThread). Caso a letra ‘s’ seja inserida antes do encerramento do programa, ocorrerá um encerramento precoce e a taxa de satisfação será calculada baseando-se nos “clientes atendidos” até o momento do fim do programa.

atendimento.c - Filas Encadeadas

São criadas duas filas encadeadas (com alternância):

- nQueue (fila normal para clientes)
- pQueue (fila de prioridade para clientes)

A thread `reception` coloca struct Cliente na fila, a thread `service` tira struct Cliente da fila.

O dequeue das filas é feito de 2:1 (2 prioridade, 1 normais)

analista.c

Em paralelo, o processo `analista` é executado:

- Gera um arquivo `pidanalista.txt`, escreve seu próprio PID nele e dorme.

Motivo: a thread `service` precisa do PID desse `analista` para acordar.

atendimento.c - Thread 2 (reception)

Início de tudo.

Cria semáforos “\sem_atend” e “\sem_block”, pois é a primeira thread a ser criada;

Cria N ou infinitos (0) processos cliente com PIDs;

Cada cliente é criado com uma prioridade aleatória de 50% alta (100% da paciência) ou baixa (50% da paciência).

cliente.c

Escreve no arquivo `demanda.txt` o tempo necessário de espera até ser atendido (dequeado pela thread `service`). Tempo definido randomicamente de 15us, 5us ou 1us.

Depois de escrever no `demanda.txt`, dorme.

atendimento.c - Thread 2 (reception)

Após o processo cliente dormir, a thread reception lê a demanda.txt.

- Cria um struct Cliente, pega o tempo no demanda.txt e atribui no Cliente.serviceTime
- Coloca esse objeto Cliente na fila (nQueue ou pQueue)

OBS: Ocorre uma iteração. Após isso, o processo volta pro slide 6.

atendimento.c - Thread 1 (service)

Após reception enfileirar todos os clientes...

A thread service passa a agir. Além de acordar, retira struct Clientes da fila. Essa thread é o “atendente”.

- Retira 1 struct Cliente de umas das filas;
- Acorda o processo cliente correspondente ao PID presente no struct Cliente retirado da fila;

cliente.c

Após acordado pelo service, o processo cliente fecha o “\sem_atend”. Semáforo fica pausado pelo tempo “timeService” do struct Cliente associado.

Depois de “ser atendido”, o processo cliente libera o “\sem_atend”.

Por fim, a thread service fecha o “\sem_block”, escreve no LNG.txt (lista de números gerados) o PID do struct Cliente atendido, e depois libera o “\sem_block”.

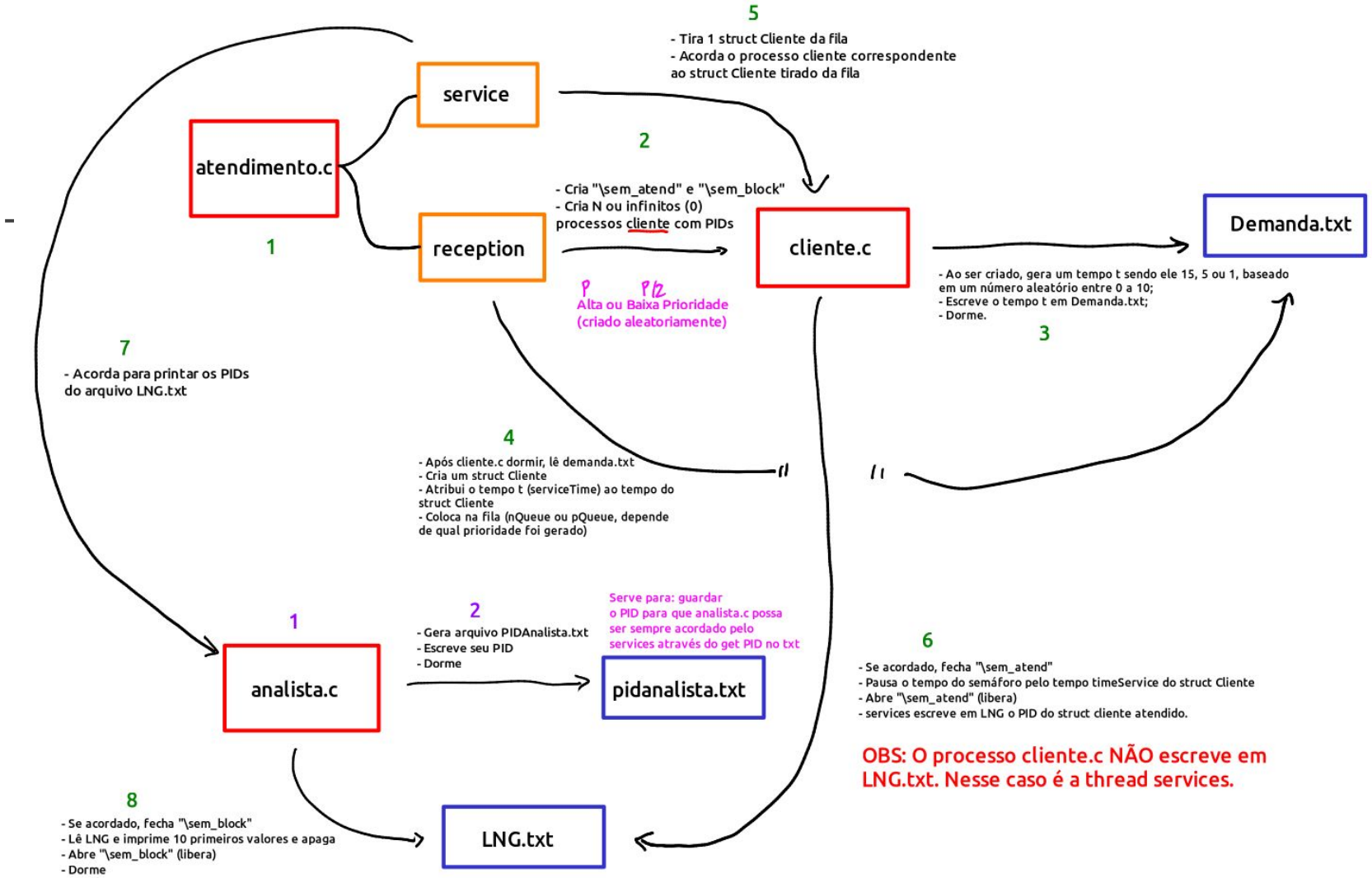
atendimento.c - Thread 1 (service)

Após todos os clientes serem retirados das filas, ocorre outro envio de sinal: a thread `service` acorda o processo `analista` (previamente adormecido no início básico `./atendimento.out` do programa).

analista.c

Com o analista acordado, ele:

- fecha o “\sem_block”;
- Lê o LNG.txt e imprime os 10 primeiros PIDs.
- Reescreve o arquivo LNG.txt sem os PIDs lidos.
- Abre o “\sem_block” (libera);
- Dorme.


5
- Tira 1 struct Cliente da fila
- Acorda o processo cliente correspondente ao struct Cliente tirado da fila
service
atendimento.c
reception
1
2
- Cria "\sem_atend" e "\sem_block"
- Cria N ou infinitos (0) processos cliente com PIDs
cliente.c
Demanda.txt
Alta ou Baixa Prioridade (criado aleatoriamente)
- Ao ser criado, gera um tempo t sendo ele 15, 5 ou 1, baseado em um número aleatório entre 0 a 10;
- Escreve o tempo t em Demanda.txt;
- Dorme.
3
7
- Acorda para printar os PIDs do arquivo LNG.txt
4
- Após cliente.c dormir, lê demanda.txt
- Cria um struct Cliente
- Atribui o tempo t (serviceTime) ao tempo do struct Cliente
- Coloca na fila (nQueue ou pQueue, depende de qual prioridade foi gerado)
1
analista.c
2
- Gera arquivo PIDAnalista.txt
- Escreve seu PID
- Dorme
pidanalista.txt
Serve para: guardar o PID para que analista.c possa ser sempre acordado pelo services através do get PID no txt
6
- Se acordado, fecha "\sem_atend"
- Pausa o tempo do semáforo pelo tempo timeService do struct Cliente
- Abre "\sem_atend" (libera)
- services escreve em LNG o PID do struct cliente atendido.
OBS: O processo cliente.c NÃO escreve em LNG.txt. Nesse caso é a thread services.
8
- Se acordado, fecha "\sem_block"
- Lê LNG e imprime 10 primeiros valores e apaga
- Abre "\sem_block" (libera)
- Dorme
LNG.txt