# Implementação de Sincronização de um Servidor de Impressão

Considere um servidor de impressão de uma empresa cujo buffer principal possui capacidade de armazenamento igual a MAX. O servidor também possui um buffer secundário, com capacidade idêntica a do principal. Dois tipos de arquivo são manipulados pelo servidor: arquivos da administração e arquivos diversos.

Entende-se que os arquivos da administração possuem maior prioridade que os arquivos diversos, de forma que esses arquivos são colocados diretamente no buffer de impressão principal. Os arquivos diversos, por sua vez, iniciam no buffer de impressão secundário. A única impressora é compartilhada pelos dois buffers, de forma que apenas um arquivo pode ser impresso por vez.

A ordem de impressão dos arquivos sempre parte do buffer principal, em que os arquivos são impressos por ordem de chegada. A cada 2 impressões de um arquivo do buffer principal, permite-se que 1 dos arquivos do buffer secundário seja impresso (caso houver). Isso evita que os arquivos diversos sofram de espera indefinida (starvation). Se não houver nenhum arquivo no buffer principal, os arquivos do buffer secundário podem ser impressos até que algum arquivo chegue no buffer principal.

A partir do momento em que um dos buffers ficar cheio, o armazenamento de arquivos fica condicionado à retirada de outros (caso em que a operação de impressão foi concluída). A geração e a impressão dos arquivos implicam em tempos de espera (valores aleatórios que devem ser gerados dinamicamente pelo programa). Cada arquivo é simulado por uma thread e possui um identificador único (id). Ao longo do *trace*, devem ser exibidas mensagens informando quais arquivos estão sendo gerados e impressos, o tipo do arquivo (administração ou diversos), os arquivos que estão esperando por vaga no buffer, bem como a situação dos buffers principal e secundário (id dos arquivos em cada buffer).

O programa recebe como parâmetros três valores (usar *argc* e *argv*): capacidade de armazenamento dos buffers (MAX), quantidade de arquivos da administração a serem impressos (QTD_ADM) e quantidade de arquivos diversos a serem impressos (QTD_DIVERSOS). Deve ser feito o teste de consistência da entrada fornecida. Após a impressão de QTD_ADM e QTD_DIVERSOS arquivos, o programa encerra.

Para evitar conflito e condições de disputa, a implementação do programa deve ser feita utilizando semáforos (primitivas sem_init, sem_wait, sem_post,…) da biblioteca *pthreads*, linguagem C, ambiente Linux. Não utilizar pthread_lock e pthread_unlock.

# Sistemas Operacionais
# Profa. Patrícia Pitthan
# Trabalho 3

**Orientações e informações importantes:**

- Trabalho individual.
- **Não** devem ser inseridos comentários no código.
- O programa deve estar modularizado (funções) e utilizar passagem de parâmetros (evitar variáveis globais quando possível).
- As dúvidas dos alunos devem se ater à especificação (o que fazer) e não a decisões de implementação (como fazer).
- Não serão analisados códigos .c antes do período de entrega. Favor não insistir.
- O arquivo .c deve ser enviado pelo Moodle. Não serão aceitos trabalhos enviados fora da plataforma Moodle.
- Data de entrega: até às **10h de 05/08/2021**. Não serão aceitos trabalhos após o prazo.
- A apresentação do trabalho será agendada posteriormente. Trabalhos não apresentados serão desconsiderados.