**TRABALHO 1 SISTEMAS OPERACIONAIS - STHREAD**

**1. Nome dos componentes do grupo e número do cartão.**

Miller Biazus - 187984
Pedro Morales - 143049

**2. Indique, para cada uma das funções que formam a biblioteca sthread, (screate, syield, swait, slock e sunlock)se as mesmas estão funcionando corretamente ou não. Para o caso de não estarem funcionando adequadamente, descrever qual é a sua visão do porquê desse não funcionamento.**

screate- funcionando
syield- funcionando
swait- funcionando
slock- funcionando
sunlock- funcionando

**3. Descreva todas as estruturas de dados utilizadas na implementação, inclusive a empregada para a variável do tipo mutex. É importante associar sua implementação com os conceitos vistos em aula.**

Estrutura das threads:

```
typedef struct tcb {
        int tid;
        int estado;
        int prio;
        int waitingFor;
        int bloqueando;
        ucontext_t *context;
} TCB;
```

Onde,
**tid** : identificador único da thread, sendo um número natural
**estado** : indica o estado da thread (apta, rodando, bloqueada)
**prio**: prioridade da thread, podendo ser de 0(prioritária) até 2 (baixa prioridade). O escalonador utiliza este critério para escalonar as thread (escalonador por prioridades)
**waitingFor**: por qual thread esta thread está esperando (inicializado com -1 e mudado para o tid da thread que chamou o swait)

**bloqueando** : qual thread está sendo bloqueada pela thread (inicializado com -1 e mudado para o tid da thread bloqueada por ela)
**context**: contexto da thread

Estrutura da lista de threads:

```
struct tList {
        TCB thread;
        struct tList *next;
};
```

onde,
**thread** : elemento thread
**next** : ponteiro para o proximo elemento da lista

Estrutura mutex:

```
typedef struct mutex {
                int flag;
                threadList* next;
}smutex_t;
```

Onde:
**flag:** indicador de livre
**next:** ponteiro para lista de threads esperando pela secao critica

**4. Liste todas as chamadas de sistema do tipo context e descreva brevemente suas operações quando uma thread chama a função screate() até que o ponto que a thread criada é inserida na lista aptos correspondente OU posta em execução (se tiver prioridade maior que a sua criadora). É importante associar sua implementação com os conceitos vistos em aula.**

A função screate() é utilizada para gerar novas threads. Ela checa se a prioridade é válida. Se não for, retorna código de erro.
Se ela for a primeira função a ser chamada ela inicializa tudo que precisa ser inicializado no sistema.

Ela inicializa o necessário ( tid, estado(Ready), prioridade) e coloca-a na fila de aptos. Incrementa-se, após, o contador do nro total de threads e do próximo valor de tid. Depois retorna o tid criado.

**5. Liste todas as chamadas de sistema do tipo context e descreva brevemente suas operações desde que uma thread chamou a função syield() até que a próxima thread seja posta em execução. É importante associar sua implementação com os conceitos vistos em aula.**

A função yield() faz com que uma thread libere voluntariamente a CPU para que outra thread seja executada.
Se ela for a primeira função a ser chamada ela inicializa tudo que precisa ser inicializado no sistema.
Quando uma thread chama a função yield, ela volta automaticamente para a fila de aptos de acordo com a sua prioridade (prioridade maior antes).
A funcao yield() então verifica se a operacao é valida, testando se existe alguma thread na lista de aptos, e se o tid é válido. Quando validada, a thread atual indica que não terminou e que seu estado passará para apto; a thread é inserida na lista de aptos, o contexto é salvo e o controle é transferido para o escalonador através de swapcontext(), que decidirá qual será a próxima thread a ser executada.

**6. Liste todas as chamadas de sistema do tipo context e descreva brevemente suas operações desde que uma thread chamou a função swait() até que a próxima thread seja posta em execução. Explicite o que acontece caso uma thread execute swait() para esperar por uma thread que já tenha encerrado. É importante associar sua implementação com os conceitos vistos em aula.**

A função swait() faz sincronização entre threads, ou seja, a thread que a chama espera até que a thread com o tid passado como argumento termine.
Se ela for a primeira função a ser chamada ela inicializa tudo que precisa ser inicializado no sistema.
Esta função verifica se já nao existe thread bloqueando a outra thread. Se não houver, modifica o estado atual, coloca a thread em blocked, e atualiza o valor de waitingFor da thread bloqueada e o valor de bloqueando da thread chamada, para que, quando esta termine, a thread chamadora seja desbloqueada e, assim, a thread chamadora seja inserida na lista de bloqueados, o contexto atual salvo e transferido o controle para o escalonador utilizando a funcao swapcontext() .

**7. Liste todas as chamadas de sistema do tipo context e descreva brevemente suas operações desde que uma thread chamou a função slock() até ela continuar em execução ou ser bloqueada. É importante associar sua implementação com os conceitos vistos em aula.**

Depois que uma thread chama a slock() é checado se a flag para a variavel mutex está liberada. Se estiver, a flag recebe sinal de bloqueio e a thread é autorizada a entrar na seção crítica. Caso contrário a thread que tentou acessar é bloqueada e o contexto atual é salvo e transferido para o escalonador utilizando a funcao swapcontext() .

**8. Liste todas as chamadas de sistema do tipo context e descreva brevemente suas operações desde que uma thread chamou a função sunlock() e tenha provocado o desbloqueio de outra thread. É importante associar sua implementação com os conceitos vistos em aula.**

A função checa se existe thread esperando para utilizar a variável. Se sim, a primeira é liberada (passa de bloqueada para apta) e a flag de uso da variável mutex é liberado (setado como livre). Se não existirem threads então a flag fica com sinal de livre, apenas, e retorna-se para a thread que liberou a variavel mutex.

**9. Qual a metodologia de teste utilizada? Isso é, quais foram os passos (e programas) efetuados para testar a sthread desenvolvida? Descreva, brevemente, o que cada um dos programas de teste entregues executa.**

Utilizei os vários exemplos disponibilizados para o trabalho, além de fazer alguns testes especificos para cada função. Utilizei os debuggers gdb e valgrind para alguns erros pontuais em caso de segmentation fault.

Também utilizo de prints que informam o estado atual das thread para cada passo do programa, o que auxilia para checar se s threads estão sendo escalonadas corretamente, o número de threads em cada fila etc.

**10. Quais as principais dificuldades encontradas e quais as soluções empregadas para contorná-las.**

Dificuldade em lidar com o tempo de implementação, pois tem muito detalhe a levar em conta. Além disso alguns erros na depuração e problemas para reproduzir erros específicios, os quais conseguimos resolver usando os debuggers.