Trabalho Prático I - Implementação da Biblioteca *Pithreads*

(1) Nome dos componentes do grupo e número de cartão

Nome	**Cartão**
Arthur Lages Lenz	218316
Vinícius Silveira Goulart	207306

(2) Resumo executivo das funções

Função	Status (OK ou Não)	Se não OK, explicação possível
picreate	OK	
piwait	OK	
piyield	OK	
pimutex_init	OK	
pilock	OK	
piunlock	OK	

(3) Testes e resultados esperados

- **mutex.c**: O teste criado para o mutex foi a criação de duas threads de mesma prioridade, onde uma possui funcionalidade de soma e a outra de subtração. A escolha de prioridades iguais, aliada da chamada de um yield a cada iteração de um laço, força a tentativa do chaveamento entre as duas threads até que o mutex criado libere uma das threads (com o uso da *piunlock*).

Assim, podemos notar que o inteiro *n* utilizado passa por todas as iterações do laço de uma função, e depois pelas iterações da segunda thread.

Com a remoção das linhas de *pilock* e *piunlock*, notamos a alternância entre as funções que operam sobre *n*.

- **create.c**: Criamos um teste para o create, simplesmente para garantir que as threads fossem criadas devidamente (alocando seus contextos, pilhas, contextos de saída, …) juntamente com uma estrutura gerada especialmente para a thread *main*, durante a primeira chamada da libpthread.

Também utilizamos o teste para garantir que a *pithread* teria o comportamento desejado de acordo com os créditos de criação passados como argumento.

(4) Principais dificuldades encontradas

- Falha de segmentação a partir de 8 threads:

Durante a execução dos testes, alguns programas geravam falha de segmentação. Após a tentativa de depuração, descobrimos que o comportamento se repetia quando 8 ou mais threads eram criadas.

Durante a execução, quando uma thread era finalizada, e o fluxo do programa era alterado para uma função "*terminateThread"*, a estrutura TCB_t alocada encontrava-se com lixo.

Com a utilização da ferramenta Valgrind, analisando leaks de memória (com *--leak-check=yes*) não conseguimos encontrar o erro (em alguns casos, o uso do Valgrind não resultava em *segfault*). É possivel que o tamanho dos frames das pilhas seja maior com o uso da ferramenta, e assim os dados não sejam sobrescritos.
Acreditamos que ao alocar 8 ou mais threads, o conteúdo de alguns atributos são sobrescritos, e seu acesso indevido gera a falha de segmentação.