

Relatório Trabalho I

Nicolas Pessutto
João Henz

Maio de 2017

Biblioteca Cthread

| Função | Foi implementada? | Está Funcionando? | Por que não está funcionando? |
|-----------|-------------------|-------------------|-------------------------------|
| ccreate | | | - |
| csetprio | | | - |
| cyield | | | - |
| cjoin | | | - |
| csem_init | | | - |
| cwait | | | - |
| csignal | | | - |
| cidentify | | | - |

Testes

foo_bar

Argumentos: nenhum

Saída Esperada: (imagem 1)

Objetivo: testar a função *cyield*, *cjoin* e *ccreate* criando duas threads simples com a mesma prioridade e as alternando.

```
FOO (i=15)
=====BAR (i=15)
FOO (i=30)
=====BAR (i=30)
FOO (i=45)
=====BAR (i=45)
FOO (i=60)
=====BAR (i=60)
FOO (i=75)
=====BAR (i=75)
FOO (i=90)
=====BAR (i=90)
```

foo_bar_testacsetprio

Argumentos: nenhum

Saída Esperada: (imagem 2)

Objetivo: usar a *csetprio* para mudar as prioridades das threads *foo* e *bar* antes e durante sua execução, neste modo testando se *csetprio* funciona com thread ativas e com threads em execução.

```
FOO (i=15)
FOO (i=30)
=====BAR (i=15)
=====BAR (i=30)
=====BAR (i=45)
=====BAR (i=60)
=====BAR (i=75)
FOO (i=45)
FOO (i=60)
FOO (i=75)
FOO (i=90)
=====BAR (i=90)
```

2

testa_semaforo

Argumentos: nenhum

3

Saída Esperada: (imagem 3)

Objetivo: testar *cwait*, *csignal* e

csem_init. Duas funções, *func0* e *func1*

querem usar o mesmo recurso: *rec_imaginario*. *Func0* é executada primeiro, então *func1* só utiliza o *rec_imaginario* depois de *func0* o liberar.

```
Inicio do testa_semaforo
func0:: pedindo recurso imaginario
func0:: usando recurso imaginario
func1:: pedindo recurso imaginario
func0:: liberando recurso imaginario
func1:: usando recurso imaginario
func1:: liberando recurso imaginario
Fim testa_semaforo
```

FINALTEST

Argumentos: roda com nenhum, mas caso se queira aumentar o *indice_de_procrastinacao* [numero de repetições do laço dos *procrastinador*], pode-se inserir um int.

Saída Esperada: (imagem 4) [execução sem entradas]

Objetivo: testar o *csetprio* para funções que estão na fila de bloqueados. Todas as funções são criadas com prioridade 0, mas a função *mestre* usa o semáforo do recurso necessário para o *acabador*, então ele aguarda sua vez novamente, mas quando sua vez chega, a função *mestre* diminuiu sua prioridade, então a função *acabador* deve aguardar todas as funções *procrastinador* terminarem sua execução...

```
Iniciando um programa que quer acabar!
mestre:: Bloqueando o hardware de acabar programas!
acabador:: Ja posso acabar?
procrastinador:: ainda nao!
mestre:: Mudando a prioridade do acabador!
mestre:: Liberando o hardware de acabar programas!
procrastinador:: ainda nao!
acabador:: Finalmente, acabando o programa!
```


