Alunos: Guilherme Souza Santos & João Pedro Costa

Relatório Projeto 01

1. Explicar o objetivo e os parâmetros de cada uma das quatro funções:

Em geral essas funções permitem trocar contextos em nível de usuário dentro de um processo.

a) getcontext(&a)

A função inicializa a estrutura apontada pelo “a” para o contexto a nível de usuário atual. O ucontext_t que é o tipo que o “a” aponta, define o contexto a nível de usuário. Caso seja bem-sucedido o processo retorna 0, senão retorna -1.

b) setcontext(&a)

Essa função restaura o contexto a nível de usuário apontado pelo “a”, a execução do contexto ocorre do momento em que a tarefa executou pela ultima vez e teve seu contexto salvo. Caso a função tenha funcionado corretamente ela não retorna nada. Para realizar essa chamada é precisa obter o contexto pela função getcontext().

c) swapcontext(&a,&b)

Permite o usuário trocar tarefas em nível de usuário, salvando o contexto atual de “a” e restaura o de “b”. Caso tenha funcionado corretamente retorna 0, senão -1.

d) makecontext(&a,...)

Essa função permite modificar o contexto da tarefa “a”, passando por parâmetro o que deve ser modificado, deve receber o contexto pela função getcontext(). Antes da chamada do makecontext, deve se garantir que o contexto a ser modificado tem o seu *stack,* alocado para realizar a mudança.

2. Explicar o significado dos campos da estrutura ucontext_t que foram utilizados no código.

ucontext_t *uc_link: é um ponteiro para o contexto, que será usada quando esse contexto for resumido por sua tarefa.

sigset_t uc_sigmask: Conjunto de sinais que são bloqueados quando o contexto está ativo.

stack_t uc_stack: a pilha usada por esse contexto;

mcontext_t uc_mcontext: uma representação específica da máquina do contexto salvo.

3. Explicar cada linha do código pingpong.c que chame uma dessas funções ou que manipule estruturas do tipo ucontext_t.

Basicamente as funções que utilizam essas manipulações e estruturas são as de criação de tarefa, troca de tarefa e termino de tarefa.

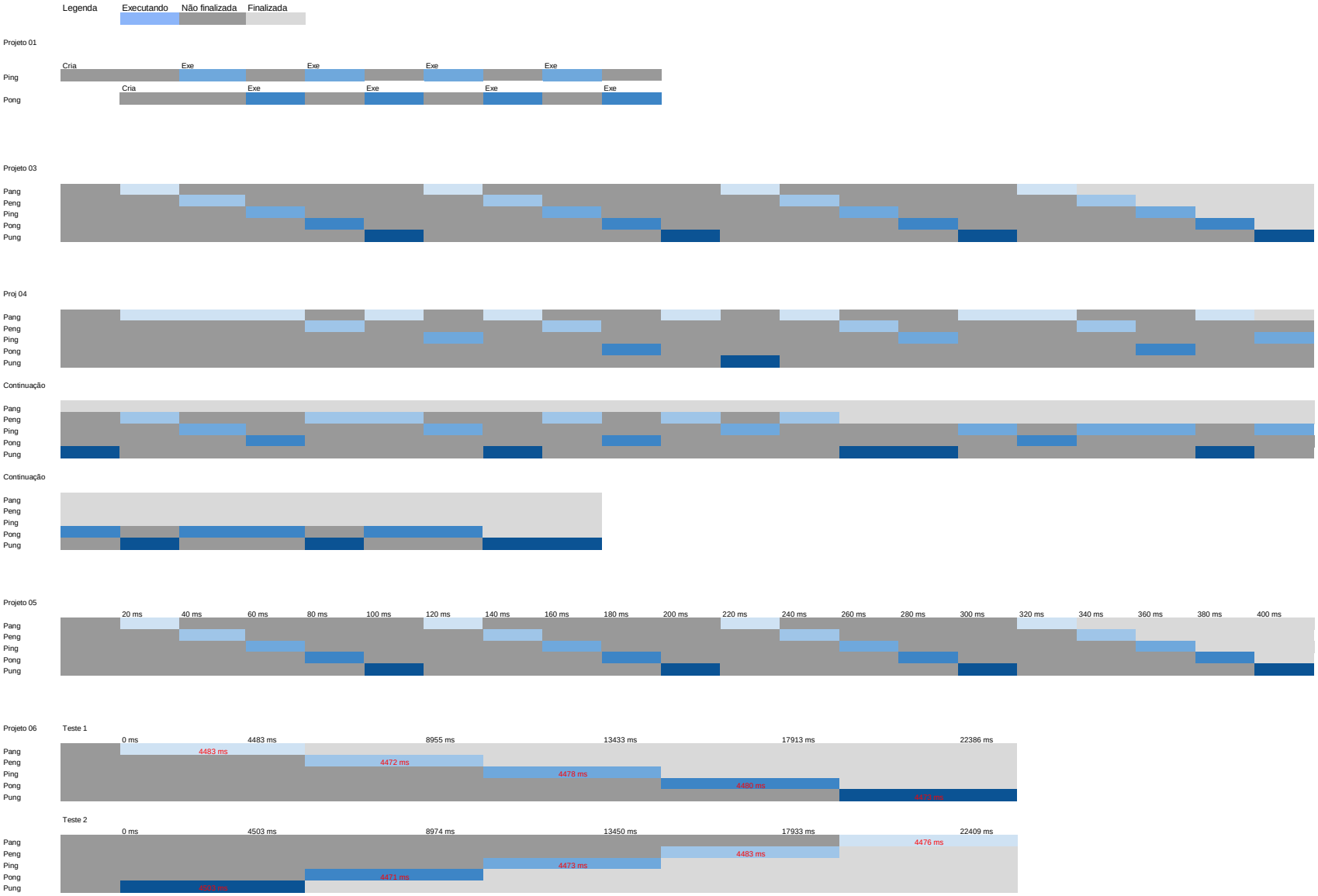
Dentro da função “task_create”, é passado como parâmetro uma tarefa a ser criada e alocada, inicia-se a função com o “getcontext” da tarefa que está no parâmetro. Então são alocados os contextos dessa tarefa, para que a função “makecontext” possa ser chamada logo em seguida. Agora que possuímos o contexto da tarefa por meio da função “getcontext” e essa tarefa foi devidamente alocada, o “makecontext” é chamado utilizando em seus parâmetros a tarefa que veio por meio do parâmetro da “task_create”.

Quando se realiza a ação de trocar uma tarefa, por meio da função “task_switch” é utilizado o “swapcontext”, com ele é salvo o contexto da tarefa que está executando no momento (chamada “taskAtual”) e essa tarefa é substituída pela tarefa que veio pelo parâmetro da função “task_switch”.

No termino de uma tarefa, durante a execução da função “task_exit”, é realizado um procedimento semelhante ao do “task_switch”. O “swapcontext” é usado para salvar o contexto da tarefa que está sendo executada no momento e para retomar a tarefa principal ou em alguns casos o “dispatcher”.

Obs: As linhas de código onde se encontram as funções citadas variam de acordo com cada parte do projeto.

4. Desenhar o diagrama de tempo da execução.


Legenda
Executando
Não finalizada
Finalizada
Projeto 01
Cria
Exe
Ping
Pong
Projeto 03
Pang
Peng
Ping
Pong
Pung
Proj 04
Continuação
Projeto 05
20 ms
40 ms
60 ms
80 ms
100 ms
120 ms
140 ms
160 ms
180 ms
200 ms
220 ms
240 ms
260 ms
280 ms
300 ms
320 ms
340 ms
360 ms
380 ms
400 ms
Projeto 06
Teste 1
0 ms
4483 ms
8955 ms
13433 ms
17913 ms
22386 ms
4472 ms
4478 ms
4480 ms
Teste 2
4503 ms
8974 ms
13450 ms
17933 ms
22409 ms
4476 ms
4473 ms
4471 ms