Nome: Pedro Henrique Belotto **Turma S73**
Ra: 1189212

Explicação do código

O código do arquivo "contexts.c", tem como objetivo apresentar como mudanças de trocas de contexto funcionam. Para isso é necessário o uso da biblioteca <ucontext.h>, que possui as funções para o manuseio de contextos dentro de um sistema operacional. Os contextos nesse código é referente a duas funções denominadas BodyPing e BodyPong.

Quando executado o código apresenta no monitor a seguinte saída:

Main INICIO
Ping iniciada
Ping 0
Pong iniciada
Pong 0
Ping 1
Pong 1
Ping 2
Pong 2
Ping 3
Pong 3
Ping FIM
Pong FIM
*Main FI*M

Para isso são utilizadas 3 funções: A função main, padrão da linguagem c e as outras duas funções associadas ao contexto, BodyPing e BodyPong, em que ambas possuem um loop que repete 4 vezes; dentro do loop ocorre impressão no monitor a mensagem passada por valor na função e a troca de contexto para a outra função.

Para inicio a função main é invocado a função getcontext, onde nela se passa o ponteiro apontando para o primeiro contexto a ser criado; nessa função é associado o contexto aquele ponteiro. Após isso são modificado os valores daquele contexto e alocado os valores de memória, para então com o uso da função makecontext(), ser realizado a criação do contexto junto a função que é passada por parâmetro nessa função. Esse processo é realizado duas vezes para as duas funções criadas.

Depois da criação dos dois contextos ocorre a troca de contexto da função main para o contexto onde esta a função BodyPing e dessa há a troca para o contexto onde esta a função BodyPong, e cada uma delas alternam entre elas até o final do loop presente na função, para que então seja feita a troca para o contexto Main e o programa seja finalizado.

Biblioteca ucontext

int getcontext(ucontext_t* context):

Essa função inicializa uma estrutura de contexto e salva na variável context passada pela função. Essa função é utilizada no início das operações que envolve contextos, a partir dela que torna possível a manipulação dos parâmetros e da execução das outras funções.

int setcontext(const ucontext_t *_ucp_);

Restaura um contexto que foi criado anteriormente por getcontext().

void makecontext(ucontext_t *_ucp_, (void *_func_)(), int _argc_, ...)

Essa função pega a estrutura criada pela função getcontext() e modifica ela com os parâmetros passados, sendo um deles uma função(void *_func)_ que representará o comportamento, e os argumentos(argc) a serem utilizados. Essa função é utilizada após criado a estrutura do contexto, e ela faz as modificações que diz a respeito ao funcionamento, como a função que aquele contexto executa.

int swapcontext(uint setcontext(const ucontext_t *_ucp_);
context_t *_oucp_, const ucontext_t *_ucp_);

Essa função, recebe dois contextos diferentes, *oucp e *ucp sendo o primeiro o contexto da tarefa que está em execução e o segundo é contexto em sera transferido o contexto. E então é salvo o contexto *oucp e é transferido a execução para o contexto *ucp. Essa Função basicamente é feita para trocas de contexto, quando por exemplo ocorrer uma interrupção da tarefa e é necessário salvar seu contexto e realizar a transferência para outra tarefa.

Pingpong

O projeto ping pong SO, tem como objetivo a criação de um sistema operacional simplificado, com o propósito educacional. Nele há utilização da biblioteca ucontext.h, a qual suas funções foram explicadas acima
Abaixo mostra a utilização dessas funções, na construção do programa pingpong.c que possui as funções do sistema.

função task_create

Essa função tem como objetivo a criação de uma tarefa, passada por parâmetro na chamada da função, além da função que será associada a esse contexto. Essa tarefa é descrita como um struct, que possui uma variável do tipo ucontext e outras variáveis, utilizadas para a coordenação de tarefas dentro do sistema operacional.

- *linha 27 getcontext(&(task->context));*
-

Nessa linha, a chamada do método getcontext(), cria a estrutura necessária para o manuseio do contexto

- linha 31 a 34
```
task->context.uc_stack.ss_sp = stack ;
task->context.uc_stack.ss_size = STACKSIZE;
task->context.uc_stack.ss_flags = 0;
task->context.uc_link = 0;
```

Esse bloco de código, aloca a memória utilizada pelo contexto e define seus parâmetros básicos .

- *linha 40 makecontext(&(task->context), (void *)(*start_func), 1, arg);*

Essa linha cria o contexto da tarefa, a partir da estrutura criada anteriormente. Para isso ela passa o contexto da variável da struct, a função recebida pela função task_create, o número de argumentos a serem utilizados por esse contexto e os argumentos, que no caso desse projeto é apenas 1.

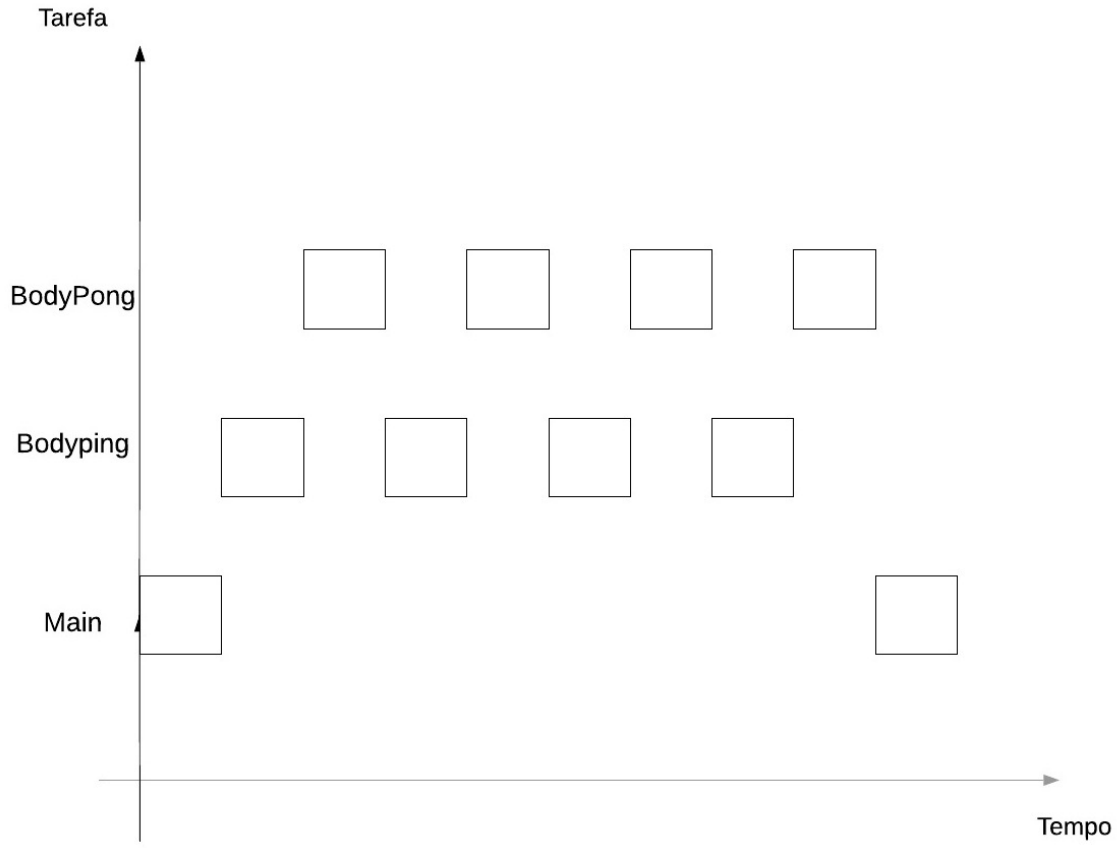
função task_switch

Essa função faz o encapsulamento da troca de contexto, recebendo como argumento a tarefa que irá assumir a execução. Vale lembrar que essa função apenas realiza a troca de tarefas, o gerenciamento, como a utilização de filas é feita por outra função.

- linha 64 swapcontext(&(ant->context), &(task->context))<0

Essa linha, que faz a troca de contexto em si, ela salva o contexto da função que está em execução e

Diagrama de tempo



Referências

https://pubs.opengroup.org/onlinepubs/7908799/xsh/ucontext.h.html