

# Relatório do Trabalho p01 - Trocas de Contexto

Aluno: Breno Moura de Abreu

## Introdução:

No projeto “p01 - Trocas de Contexto”, os alunos receberam um código de demonstração do funcionamento das trocas de contexto de um sistema operacional, utilizando a biblioteca `ucontext.h`, que permite a utilização do registro `ucontext_t` e quatro funções que permitem manipulá-la. São elas: `getcontext`, `setcontext`, `makecontext` e `swapcontext`; que serão explicadas posteriormente.

O código em si gera um programa que alterna o contexto entre três funções: a `main`, a `BodyPing` e a `BodyPong`, que irão simplesmente imprimir textos na tela dependendo de onde o contexto atuante está. Começando pela função `main`, que irá criar os contextos inicialmente, e então muda-o para a função `BodyPing`, que alterna para a função `BodyPong`. Esse último processo é repetido quatro vezes, e então é retornado para a função `main`, que finaliza o programa.

## Questão 1: Explicar o objetivo e os parâmetros de cada uma das quatro funções citadas acima

O “`ucontext_t`”, é uma struct pertencente à biblioteca `ucontext.h`. O ‘u’ significa user, portanto “`ucontext_t`” significa user context. Essa struct é um contexto que irá guardar as informações necessárias para que uma tarefa possa ser interrompida e futuramente continuada a partir do mesmo ponto. É necessária para a execução de processos e threads que irão executar múltiplas tarefas concomitantes.

A função “`int getcontext(ucontext_t *ucp)`” recebe um ponteiro para uma struct “`ucontext_t`”. A struct será “`ucp`” inicializada e o contexto atual será salvo nela. Se bem sucedida, retorna 0, caso contrário retorna -1.

A função “`int setcontext(const ucontext_t *ucp)`” recebe um ponteiro para uma struct “`ucontext_t`”. Restaura o contexto apontado pela struct recebida. Se bem sucedida, não retorna, caso contrário retorna -1.

A função “`void makecontext(ucontext_t *ucp, void (*func)(), int argc, ...)`” manipula contextos criados pelo usuário modificando um contexto apontado por `ucp`. Antes da função ser chamada, deve-se alocar uma nova pilha para esse contexto, mudar seu endereço para “`ucp->uc_stack`”, e definir um contexto que irá sucedê-lo, mudando o endereço deste último para “`ucp->us_link`”. Quando o contexto `ucp` é chamado a função `func` é ativada e passada a ela um conjunto de argumentos recebidos também na função `makecontext`.

A função “`int swapcontext(ucontext_t *oucp, const ucontext_t *ucp)`” manipula contextos criados pelo usuário salvando o contexto atual em “`oucp`”, e ativando o contexto apontado por “`ucp`”. Quando bem sucedido a função não retorna, caso contrário retorna -1.

## Questão 2: Explicar o significado dos campos da estrutura “ucontext\_t” que foram utilizados no código

Campo “uc\_stack.ss\_sp” - é um ponteiro para a pilha que será usada nesse contexto.

Campo “uc\_stack.ss\_size” - determina o tamanho da pilha.

Campo “uc\_stack.ss\_flags” - determina flags usadas na pilha.

Campo “uc\_link” - caso o contexto atual retorne, aqui será armazenado o endereço para o próximo contexto que será executado.

## Questão 3: Explicar cada linha do código de “pingpong.c” que chame uma dessas funções ou que manipule estruturas do tipo “ucontext\_t”

Referente ao arquivo pingpong.c do projeto p06.

Na função task\_create(...), uma tarefa recebida é inicializada utilizando a função getcontext(), como apontado na questão 1, antes de chamar a função makecontext() é necessário modificar certos membros da struct e alocar uma nova pilha a esse contexto, preenchendo os campos definidos na questão 2, além de apontar o contexto que irá sucedê-lo, que nesse caso é um ponteiro nulo. A função makecontext() é acionada, conectando a tarefa com a função recebida, além de passar uma variável do tipo void que será utilizada na função recebida.

Na função task\_switch(...) é utilizada a função swapcontext(), que troca o contexto da tarefa atual para a tarefa recebida.

## Questão 4: Desenhar o diagrama de tempo da execução:

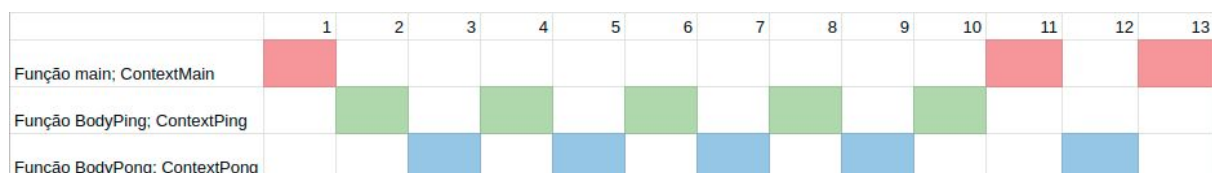


Figura 1. Diagrama de Tempo de Execução dos contextos do programa gerado pelo arquivo “contexts.c”.