RELATORIO

SISTEMAS OPERACIONAIS - Projeto 01
ALUNO - Rafael Garcia de Carvalho

1- Explicar o objetivo e os parâmetros das funções:

-> getContext(&context)

Salva o processo atual em uma estrutura do tipo ‘ucontext_t’ representada pela variável ‘context’.

Pode ser utilizada antes de uma operação de troca de contexto, já que o mesmo será salvo na variável passada por parâmetro e pode ser restaurada posteriormente.

-> setContext(&context)

Troca imediatamente o contexto atual para o contexto apontado pela variável ‘context’ do tipo ‘ucontext_t’.

-> swapContext(&current, &another)

Salva o contexto atual na variável ‘current’ e então troca o contexto usado durante o fluxo de execução para o que está contido na variável ‘another’.

Ambas as variáveis passadas como parâmetro para esta função são do tipo ‘ucontext_t’.

-> makeContext(&context, func(), arg_qnt, args)

Opera mudanças no contexto apontado pela variável ‘context’. Dentro desse contexto são salvos:

- Uma função que será executada durante a chamada a esse contexto
- A quantidade de argumentos que a função mencionada acima deve receber
- Os argumentos de fato, ordenados corretamente

A execução dessa função é acionada quando uma operação de swap ou set é utilizada.

2 - Explicar o significado de cada um dos campos da estrutura ucontext_t que foram utilizados no código:

- uc_stack: Informações da stack que será usada pela função alocada no contexto
- uc_stack.ss_sp: Ponteiro para o início da stack mencionada acima
- uc_stack.ss_size: Tamanho alocado para a stack, informado em bytes
- uc_link: O próximo contexto que será utilizado durante o fluxo de execução depois que todas as operações especificadas no contexto corrente terminem.

3 - Explicar em detalhes o funcionamento do código do arquivo contexts.c:

-> No inicio da funcao main() temos um print da string “Main INICIO” que evidencia o começo da execução do código.

-> Chamada a função getContext() que salva o contexto atual na variável ‘contextPing’

-> Alocando uma stack de tamanho constante (definido por STACKSIZE)

-> Caso a stack tenha sido alocada sem problemas: Guarda-se no contexto que essa stack deve ser utilizada, o tamanho dessa stack, define que não existem flags e que não se estabeleceu nenhum processo sucessor após o término da execução de um processo que utilize esse contexto.

-> Uma chamada a função makeContext(), indicando que a execução de uma chamada a swapContext() ou setContext() que passe como parâmetro contextPing deve trigar a execução do código contido em bodyPing() passando somente 1 parâmetro, a string “PING”

-> Chamada a função getContext() passando por parâmetro a variável ‘contextPong’. Isso inicializa uma estruta do tipo ucontext_t e aponta ‘contextPong’ para a mesma.

-> Alocando uma stack de tamanho constante (definido por STACKSIZE)

-> Caso a stack tenha sido alocada sem problemas: Guarda-se no contexto que essa stack deve ser utilizada, o tamanho dessa stack, define que não existem flags e que não se estabeleceu nenhum processo sucessor após o término da execução de um processo que utilize esse contexto.

-> Uma chamada a função makeContext(), indicando que a execução de uma chamada a swapContext() ou setContext() que passe como parâmetro ‘contextPong’ deve trigar a execução do código contido em bodyPong() passando somente 1 parâmetro, a string “PONG”

-> Um chamada a função swapContext(). O contexto atual de mainContext() é salvo e então deve ser trocado por ‘contextPing’. O contexto de ‘contextPing’ é carregado na memória e ,como definido anteriormente, é executado a funcao bodyPing()

-> bodyPing() executa uma chamada a swapContext() que salva o contexto atual de ‘contextPing’ e troca o contexto para ‘contextPong’. Como definido anteriormente isso triga a execução de bodyPong().

-> bodyPong() executa uma chamada a swapContext() que salva o contexto atual em ‘contextPong’ e troca o contexto para ‘contextPing’. Como definido anteriormente isso triga a execução de bodyPing()

As chamadas a função swapContext() e consequentemente a execução de bodyPing() e bodyPong() serão realizadas em loop uma quantidade de vezes igual ao valor da variável ‘i’ contida em cada uma delas

4 - Diagrama de tempo de execução: