Relatório de execução do programa "context"

Guilherme Morais Lopes dos Santos - GRR 20163043

1) Explicar o objetivo e os parâmetros de cada uma das quatro funções acima.
    a) getcontext(&a): recebe como parâmetro uma variável do tipo ucontext_t e salva o contexto atual nela.

    b) setcontext(&a): restaura o contexto salvo anteriormente na variável passada como parâmetro.

    c) swapcontext(&a, &b): troca de contexto, salvando a e restaurando b;

    d) makecontext(ucontext_t *a, void (*func)(), int argc, ...): recebe como parâmetro uma variável a com um contexto salvo; se assegura de que func será chamada quando houver um swapcontext ou setcontext.

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

```
typedef struct ucontext_t
  {
    unsigned long int __ctx(uc_flags);
    struct ucontext_t *uc_link;
    stack_t uc_stack;
    mcontext_t uc_mcontext;
    sigset_t uc_sigmask;
    struct _libc_fpstate __fpregs_mem;
  } ucontext_t;
```

    a) unsigned long int __ctx(uc_flags)
    b) struct ucontext_t *uc_link
    c) stack_t uc_sigmask
    d) struct _libc_fpstate __fpregs_mem

3) Explicar cada linha do código de pingpong.c que chame uma dessas funções ou que manipule estruturas do tipo ucontext_t
    a) Linhas que chamam umas das funções
        i) getcontext
            (1) getcontext (&ContextPing): salva o contexto atual na variável ContextPing do tipo ucontext_t;
            (2) getcontext (&ContextPong): salva o contexto atual na variável ContextPong do tipo ucontext_t;
        ii) swapcontext
            (1) swapcontext (&ContextPing, &ContextPong): salva o contexto de ContextPing e restaura o contexto de ContextPong;

(2) swapcontext (&ContextPing, &ContextMain): salva o contexto de ContextPing e restaura o contexto de ContextMain;
(3) swapcontext (&ContextMain, &ContextPing): salva o contexto atual (da Main) na variável ContextMain e em seguida restaura o contexto de ContextPing. ContextPing por sua vez executa a função BodyPing, que foi passada como parâmetro para makecontext quando esta “montou” o contexto ContextPing.

iii) makecontext
(1) makecontext (&ContextPing, (void*)(*BodyPing), 1, " Ping"): se assegura de que o contexto armazenado em ContextPing execute a função BodyPing com um parâmetro (a string que precede). O mesmo é feito para CoontextPong posteriormente.

4) Desenhar o diagrama de tempo da execução.
i) **Main** getcontext(&ContextPing);
ii) **Main** makecontext(&ContextPing);
iii) **Main** getcontext(&ContextPong);
iv) **Main** makecontext(&ContextPong);
v) **Main** swapcontext(&ContextMain, &ContextPing)
(1) **BodyPing** print(“Ping: início”)
(2) **BodyPing** print(“Ping: 0”)
(3) **BodyPing** swapcontext(&ContextPing, &ContextPong)
(a) **BodyPong** print(“Pong: início”)
(b) **BodyPong** print(“Pong: 0”)
(c) **BodyPong** swapcontext(&ContextPong, &ContextPing)
(4) **BodyPing** print(“Ping: 1”)
(5) **BodyPing** swapcontext(&ContextPing, &ContextPong)
(a) **BodyPong** print(“Pong: 1”)
(b) **BodyPong** swapcontext(&ContextPong, &ContextPing)
(6) **BodyPing** print(“Ping: 2”)
(7) **BodyPing** swapcontext(&ContextPing, &ContextPong)
(a) **BodyPong** print(“Pong: 2”)
(b) **BodyPong** swapcontext(&ContextPong, &ContextPing)
(8) **BodyPing** print(“Ping: 3”)
(9) **BodyPing** swapcontext(&ContextPing, &ContextPong)
(a) **BodyPong** print(“Pong: 3”)
(b) **BodyPong** swapcontext(&ContextPong, &ContextPing)
(10) **BodyPing** print(“Ping: fim”)
(11) **BodyPing** swapcontext(&ContextPing, &ContextMain)
vi) **Main** swapcontext(&ContextMain, &ContextPong)
(a) **BodyPong** print(“Pong: fim”)

(b) **BodyPong** swapcontext(&ContextPong, &ContextMain)

vii) **Main** printf("main: fim\n") ;