1. Parâmetros
   1. getcontext(ucontext\_t \*ucp) - função responsável por inicializar a variável a do tipo ucontext\_t, utilizada para armazenar informações e referências a contextos, a partir do contexto ativo naquele momento
   2. setcontext(ucontext\_t \*ucp) - a função recebe uma variável do tipo ucontext\_t e tem por objetivo restaurar o contexto referenciado através desta variável, que foi inicializada através de uma chamada de getcontext() ou makecontext.
   3. swapcontext(ucontext\_t \*ucp, ucontext\_t \*ucp) - salva o contexto atual na primeira variável a do tipo ucontext\_t e ativa o contexto referenciado na segunda variável b do tipo ucontext\_t
   4. makecontext(ucontext\_t \*ucp, void (\*func)(), int argc, ...) - Altera o contexto apontado pela variável a, obtido através de uma chamada da função getcontext(). Para que isto ocorra uma pilha deve ser alocada e assinalada para o atributo uc stack e um contexto sucessor deve ser assinalado para uc\_link. Uma vez ativado este contexto a função func é chamada, passando seus n argumentos contados através de argc. Uma vez retornada a função o contexto sucessor é ativado, e caso seja nulo a thread é encerrada
   5. uc\_stack.ss\_sp - Aponta para a base da região de memória alocada para a pilha
   6. uc\_stack.ss\_size - Indica o tamanho da região alocada para a pilha
   7. uc\_stack.ss\_flags = 0;
   8. uc\_link - Refere-se ao contexto sucessor que sera ativado uma vez que a função referente ao contexto atual retornar
   9. getcontext (&ContextPing); - Ping é incializado com contexto atual
   10. ContextPing.uc\_stack.ss\_sp = stack ;

ContextPing.uc\_stack.ss\_size = STACKSIZE;

ContextPing.uc\_stack.ss\_flags = 0;

ContextPing.uc\_link = 0;

- As variavies relativas a pilha de ping são incializadas

* 1. makecontext (&ContextPing, (void\*)(\*BodyPing), 1, " Ping"); - O contexto de ping é alterado e a função BodyPing é designada para ser executada uma vez que ping for ativado, recebendo 1 parametro = " Ping"
  2. getcontext (&ContextPing); - Ping é incializado com contexto atual
  3. ContextPong.uc\_stack.ss\_sp = stack ;

ContextPong.uc\_stack.ss\_size = STACKSIZE;

ContextPong.uc\_stack.ss\_flags = 0;

ContextPong.uc\_link = 0;

- As variavies relativas a pilha de ping são incializadas

* 1. makecontext (&ContextPong, (void\*)(\*BodyPong), 1, " Pong"); - O contexto de ping é alterado e a função BodyPong é designada para ser executada uma vez que ping for ativado, recebendo 1 parametro = " Pong"
  2. swapcontext (&ContextMain, &ContextPing); - O contexto atual é salvo na variável ContextMain e o contexto de ContextPing é ativado
  3. swapcontext (&ContextMain, &ContextPong); O contexto atual é salvo na variável ContextMain e o contexto de ContextPong é ativado
  4. swapcontext (&ContextPing, &ContextPong); - O contexto atual é salvo na variável ContextPing e o contexto de ContextPong é ativado
  5. swapcontext (&ContextPong, &ContextPing); - O contexto atual é salvo na variável ContextPong e o contexto de ContextPing é ativado
  6. swapcontext (&ContextPing, &ContextMain) ; - O contexto atual é salvo na variável ContextPing e o contexto de ContextMain é ativado
  7. swapcontext (&ContextPong, &ContextMain) ; - O contexto atual é salvo na variável ContextPong e o contexto de ContextMain é ativado