Pedro Francescon Cittolin
Viviane Lima Bonfim

Projeto 1

Trocas de contexto

- **ucontext_t:** é uma struct definida com, no mínimo, um ponteiro para o próximo contexto (*uc_link), um conjunto de sinais que são bloqueados quando o contexto atual é utilizado (uc_sigmask), uma pilha para ser usada por este contexto (uc_stack) e uma variável que contem o estado atual do processo (uc_mcontext).

- **getcontext**(ucontext_t *ucp)**:** inicializa a variável apontada por *ucp com o contexto da thread que está chamando a função. O contexto é definido pelos registradores, a máscara de sinal e a pilha atual utilizada.

- **makecontext**(ucontext_t *ucp, void (*func) (void), int argc, …)**:** o parâmetro *ucp deve ter sido anteriormente inicializado por uma chamada da função "getcontext". Se o contexto de *ucp for resumido ele vai começar chamado a função *func, passando argc como argumento desta função. Antes da chamada dessa função, os argumentos uc_stack e uc_link da variável ucp deve ser inicializados.

- **setcontext** (const ucontext_t *ucp)**:** restaura o contexto apontado pela variável ucp. Se o contexto foi criado pela chamada de função “getcontext”, a execução resume com os registradores preenchidos com os mesmos valores e a mesma pilha da forma como a chamada da função “getcontext” retornou. Se o contexto foi modificado por uma chamada de função “makecontext”, a execução continua com a função passada como argumento em “makecontext”. Se essa função retorna, o programa continua sendo executado no contexto definido pela variável uc_link. Se uc_link for um ponteiro nulo, a aplicação termina normalmente com o valor do status de saída EXIT_SUCCESS.

- **swapcontext** (ucontext_t *restrict oucp, const ucontext_t *restrict ucp)**:** similar a função “setcontext”, mas ao invés de apenas substituir o contexto atual ela o salva no objeto apontado por “restrict oucp”, como se fosse uma chada de “makecontext”. O contexto atual é então resumido após a chamada dessa função.