INE 5412 Sistemas Operacionais I - 2011/1

Trabalho Laboratório 4 - Data de entrega – 04/05/2011

Exercício: [Unix, C, Pthreads]

Descrição do Programa:

Programa **main()** utiliza **pthread_create()** para criar tantas threads quantas especificadas em num_threads(que pode ser um parâmetro de entrada do **main()**). Antes de criar as threads o **main()** mostra na tela mensagem com o seu *PID* (**getpid()**) e *PPID* (**getppid()**).
Após, o **main()** espera pelo término das threads usando **pthread_join()** e mostra a mensagem/valor retornado pelas threads. Na rotina thread, é mostrado a mensagem passada pelo **main()** e também é mostrado o valor da variável global (inicializada em 0 pelo **main()**), que varia de 1 a num_threads. Finalmente, a rotina usa **pthread_exit()** para retornar um inteiro para o **main()**. O valor do inteiro é 2 vezes o *ID* da thread (**pthread_self**). Durante a mudança/mostra da variável global, **pthread_yield()** é usada para fazer com que a thread corrente (**main()**) deixe a execução para outra thread.

EX:

Esqueleto do main()

```
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <sys/types.h>
#include <unistd.h>
#include <pthread.h>    /*  Pthread library */
#define num_thread 4

void *thread_rotina(void *);  /* prototipo da rotina */
int sharedVar;                /* variável global */

int main ()
{
    void *pRtnValue;          /* ponteiro para void */
    thread_t tid[thread_num]; /* thread id */
    thread_t departed;        /* término */

    /* strings passados para as threads */
    int iValue[num_thread];
    int i; /* contador de threads */
    sharedVar = 0;

    .

    /* cria threads passando argumentos */
    for (i = 0; i < num_thread; i++)
    {
```

```
    .
    }

    /* muda variável global …. */
    for (i = 0; i < num_thread; i++)
    {
    .

    }

    /* espera término de threads */
    for (i = 0; i < thread_num; i++)
    {
    .
    .
    .

/* mostra ID da thread com valor de retorno */

        }
        else
        {
            perror("pthread_join() falhou.");
            exit(1);
        }
    }

    return(0);
}
```

Esqueleto rotina

```
void *thread_routine (void *arg)
{

    /* mostra  ID do processo */

    /* mostra valor passado */

    for (i = 0; i < num_thread; i++)
    {
        /* mostra valor da variável global */
    }

    /* mostra valor de retorno */

    /* termina thread com argumento */

}
```

Descrição das funções Pthreads em:
http://www.llnl.gov/computing/tutorials/pthreads/
http://www.yolinux.com/TUTORIALS/LinuxTutorialPosixThreads.html
http://www.inf.ufsc.br/~fernando/ine5355/programas/pthreads

Mostra do Resultado do Programa

```
processo main() PID= 10676. e  PPID=10667.
thread 4  processo ID= 10676 e PPID=10667.
thread 4 recebeu mensagem do main()=  0.
thread 4: Var Global é 1.
thread 5  processo ID= 10676 e PPID=10667.
thread 5 recebeu mensagem do main()=  1.
thread 5: Var Global é 1.
thread 6  processo ID= 10676 e PPID=10667.
thread 6 recebeu mensagem do main()=  2.
thread 6: Var Global é 1.
thread 7  processo ID= 10676 e PPID=10667.
thread 7 recebeu mensagem do main()=  3.
thread 7: Var Global é 1.
thread 4: Var Global é 2.
thread 5: Var Global é 2.
thread 6: Var Global é 2.
thread 7: Var Global é 2.
thread 4: Var Global é 3.
thread 5: Var Global é 3.
thread 6: Var Global é 3.
thread 7: Var Global é 3.
thread 4: Var Global é 4.
thread 5: Var Global é 4.
thread 6: Var Global é 4.
thread 7: Var Global é 4.
thread 4 retornará inteiro = 8.
thread 5 retornará inteiro = 10.
thread 6 retornará inteiro = 12.
thread 7 retornará inteiro = 14.
main() join com thread 7 valor de retorno 14.
main() join com thread 6 valor de retorno 12.
main() join com thread 5 valor de retorno 10.
main() join com thread 4 valor de retorno 8.
```