Licenciatura em Engenharia Informática
Sistemas Operativos 2021/2022
Trabalho Teórico-prático - threads

Objectivos

• Familiarização com o ambiente de desenvolvimento utilizado nas aulas (Unix)
• Programação em Unix (linguagem C) utilizando chamadas ao sistema (*system calls*)
• Comunicação e Sincronização de Threads

Normas e orientações para a realização dos trabalhos:

a) O trabalho deve ser realizado em linguagem C (C para UNIX) por um ou dois alunos por grupo
b) Para a resolução dos problemas deverá ler cuidadosamente o respectivo enunciado e bibliografia recomendada na disciplina.
c) Data-limite para a entrega do trabalho (Unix Programming): **16/06/2022**.
d) Para complementar a avaliação, o docente da disciplina irá solicitar aos elementos, uma apresentação do trabalho desenvolvido, onde serão discutidas as soluções encontradas.
e) A apresentação de programas, códigos, relatórios copiados implica a anulação dos mesmos e a impossibilidade de concluir a disciplina.
f) Os elementos a entregar devem consistir numa impressão **do código comentado**
- uma descrição resumida do seu funcionamento
- um (ou mais) fluxograma(s) do algoritmo(s) implementado(s)
- e envio desse código+relatório para o site da disciplina **em https://doctrino.ipt.pt**

Bibliografia Aconselhada

- Unix Systems Programming: Communication, Concurrency, and Threads (Second edition to Practical Unix Programming.), K. A. Robbins and S. Robbins, Prentice Hall, 2003
- UNIX Network Programming, Volume 2, Second Edition: Interprocess Communications, Prentice Hall, 1999, ISBN 0-13-081081-9.
- "The C Programming Language", Brian Kernighan and Denis Ritchie, PRT Prentice Hall 1988
- "Advanced Linux Programming", Copyright © 2001 by New Riders Publishing, FIRST EDITION: June, 2001 (pdf)
- Folha com os comandos básicos de Unix
- WebSite da Disciplina (SO)
  - Apontamentos de programação C para UNIX (Processes, Threads, Inter-process communication (IPC), Sockets)
    (**ver Mutexes: Locking and Unlocking; Condition Variables: waiting and signaling**)
    (ver a partir da pag. 62 a evolução dos exemplos *prodcons1.c* … *prodcons7.c*)
  - http://orion.ipt.pt/~so/pratica/programar_c.txt
  - Programming in C - UNIX System Calls and Subroutines using C, A. D. Marshall 1994-9 (*http://www.cs.cf.ac.uk/Dave/C/CE.html*)
  - The Linux Programmer's Guide (*http://www.tldp.org/LDP/lpg/*)
  - Advanced Linux Programming (local copy - book.pdf)

Exercício - Threads Unix Programming

Usando a linguagem C, as chamadas ao sistema UNIX e à biblioteca de Posix Threads implemente o seguinte esquema de programação concorrente:

Escreva um programa que crie um processo que por sua vez crie 3 threads:

- Crie 3 buffers para conter números inteiros:
  - o **buffer_in** deverá conter 20000 elementos e ser inicializado de 1 a 20000
  - o **buffer_circ** deverá suportar até 50 elementos
  - o **buffer_out** suporta 19998 elementos
- As threads produtoras, **PM_T1, PM_T2** devem preencher em simultâneo o buffer circular de 50 elementos (**buffer_circ**) com o resultado da média de 3 elementos consecutivos do buffer_in
  - **media[ ]= (buf_in[i] + buf_in[i+1]+buf_in[i+2])/3**

- A thread **CM_T** (consumidora) deve percorrer o buffer circular, e copiar esses elementos para o buffer_out
- As threads devem funcionar o mais concorrentemente possível.
- A thread **CM_T (consumidora)** deve arrancar logo que haja elementos novos no buffer (copiar e imprimir os valores de resultado do **buffer_out**)
- As threads **PM_T1, PM_T2** poderão voltar a colocar novos valores no buffer circular logo que estes tiverem sido lidos e processados pela consumidora.
- As threads **PM_T1, PM_T2** poderão avançar a uma a frente de outra de forma indiferenciada ao longo do tempo
- No final, o processo que criou as threads **PM_T1, PM_T2** e **CM_T,** deve apresentar quantas vezes essas threads produziram e consumiram elementos de forma individual.
- Não deverá existir "busy waiting" e o programa deve terminar libertando todos os recursos utilizados.
- (fluxograma, estrutura e detalhes importantes, objectivo: implementação prática!!!)

*Utilize os mecanismos **Mutexes** (Locking and Unlocking) e/ou **Condition Variables** (waiting and signaling) para sincronização de threads*

Advanced Linux Programming (local copy - book.pdf)

Para um i de 0…19999, o índex do buffer circular pode ser realizado com index= i % 50 index={0..49}

(não utilize o valor da célula dos arrays, para decidir se pode escrever ou não !!!)

Exemplo de uma possível situação que poderá ocorrer.