# Sistemas Operativos, 3.º MIEIC 2009/10, FEUP

– RMA, JVV, HSF –

November 8, 2009

## 2.º Trabalho Prático: SO Shell Stats (*sosh_s*)

### Objectivos

Completando com sucesso todas as fases deste trabalho, os alunos demonstram conhecer e saber utilizar a interface programática de UNIX para:

- criar programas multithread;
- promover a intercomunicação entre processos através de canais com nome (*named pipes*);
- evitar conflitos entre entidades concorrentes, por via de mecanismos de sincronização.

### Descrição

Pretende-se obter uma aplicação do tipo cliente-servidor capaz de lidar com situações de conflito no acesso a zonas (recursos) partilhadas. A aplicação cliente será a *sosh*, desenvolvida no trabalho prático 1, que será estendida para comunicar com um servidor (*freq*). A aplicação servidor não será interactiva (toda a comunicação com ela deve ser feita via comandos da *sosh*), e tem como objectivo guardar informação necessária para o cálculo da análise de frequência de letras. Este é um método simples para decifrar mensagens criptografadas por meio da análise, no texto criptografado, de padrões que se repetem constantemente[1]. O trabalho a desenvolver consiste nos seguintes pontos:

1. Estenda a sosh implementada no trabalho prático anterior para enviar para um canal com nome (*named pipe*), /tmp/sosh.canal, todos os comandos introduzidos pelo utilizador bem como os resultados dos mesmos. Deverá criar dois processos: um para executar o comando e outro para comunicar com o *named pipe*;

[1] `http://www.numaboa.com/criptografia/criptoanalise/310-Frequencia-no-Portugues`

2. Desenvolva um programa *freq* que lê do canal com nome /tmp/sosh.canal uma linha de texto (use o código disponibilizado no anexo A para ler uma linha de um descritor de ficheiro) e actualize a estrutura de dados com a frequência das letras. O programa deverá aceitar, através da linha de comandos, a opção '-v' que fará com que envie para o ecrã mensagens sempre que iniciar uma operação. Deverá ainda imprimir a lista de frequências após serem actualizadas;

   Sugestão:

   - Não distinga entre letras maíusculas e minúsculas, ou seja, *freq* não é case sensitive;
   - A estrutura de dados deverá ser um array de inteiros;
   - Para efeitos do cálculo das frequências, considere somente as letras do alfabeto latino, ignorando todas as outras (ver `http://pt.wikipedia.org/wiki/Alfabeto_latino`).

3. Desenvolva o comando *stats* que deverá listar em ordem crescente as frequências das letras até ao momento. O resultado a ser impresso no ecrã deverá ser enviado ao cliente (*sosh_s*) pelo servidor (*freq*) através do *named pipe* /tmp/sosh.results. Sendo que pretende as frequências até ao momento em que o comando foi invocado, não é permitido alterar os valores das estruturas de dados durante a computação do ranking de frequências.

   Sugestão:

   - Para facilitar a distinção entre o que é um comando e texto normal, crie um *named pipe* /tmp/sosh.cmd para enviar comandos ao servidor. O servidor poderá possuir um processo (ou thread) para ler dos *named pipes* /tmp/sosh.canal e /tmp/sosh.cmd.

4. Altere o programa *freq* por forma a conseguir processar do *named pipe* várias linhas de texto ao mesmo tempo (usando pthreads). O número de threads a ser criado deverá ser passado ao programa através da linha de comandos ("-t <num>"). Cada linha no *named pipe* só deverá ser processada uma única vez.

5. Desenvolva o comando *realtime <on/off>* que deverá notificar o utilizador da *sosh_s* que o top 5 de letras mais frequentes sofreu uma alteração. Naturalmente, se o *realtime* estiver off (estado por defeito) o utilizador não será informado.

   Sugestão:

   - Use o *named pipe* /tmp/sosh.results para a comunicação do top 5 se este sofrer alguma alteração. Crie um processo que leia da *named pipe* /tmp/sosh.results o resultado e imprima no ecrã (usando *unnamed pipes*).

## Valorização

Quem desejar obter o máximo da classificação estabelecida para este trabalho poderá estender *sosh_s* (versão b) por forma que possa suportar as seguintes opções

- Dotar a *sosh* (e se necessário *freq*) com os mecanismos necessários para que seja possível recolher informação das várias shells;
- *s_hist [n]* - crie e apresente no ecrã um histograma com a frequência das diferentes letras; se o for invocado sem argumentos, deverá apresentar o histograma de todas as letras, caso contrário somente o top n.

Aquando da geração do histograma, nenhum processo poderá aceder à estrutura de dados. Faça uso da biblioteca *Histograms* da *GNU Scientific Library* ou do *gnuplot* para visualizar os histogramas.

## Produto final

O trabalho total consiste na produção de ficheiro *.tgz* incluindo o código fonte, um makefile preparado para facilitar a geração do executável, e um ficheiro de texto README com os passos necessários para compilar e executar o código. Deverão também de entregar um breve relatório (inclua no relatório um esquema exemplificativo dos canais de comunicação e acessos as estrutura de dados) em formato pdf. O compacto, identificado com um nome do tipo TPnGX.tgz, onde n é o número do turno prático que estão inscritos e X é o número atribuído ao grupo pelo docente das teórico-práticas (e.g. TP1G2.tgz), deve ser entregue usando as facilidades disponíveis para a unidade curricular no moodle (`http://moodle.fe.up.pt/0910/`) até ao final do dia **13/12/2009**.

## Avaliação

A classificação do trabalho será repartida em 90% para o programa normal e 10% para a parte de valorização. O programa normal será classificado relativamente aos seguintes aspectos: bfprogramação, incluindo estrutura, comentários, funcionalidades, etc. (60%); **relatório curto**, que deve consistir na apresentação dos objectivos, estrutura trabalho, testes, conclusão e declaração de autoria (20%); **apresentação**, brevemente efectuada pelos autores ao docente que apreciará, ele mesmo, na altura, as facilidades de compilação, utilização, etc. (20%).

## Código de Honra

Espera-se que cada grupo obtenha a sua nota baseada exclusivamente no esforço e trabalho dos elementos do grupo. Consequentemente qualquer forma de plágio, constitui uma fraude inaceitável. Estes comportamentos não serão tolerados e os alunos serão punidos de acordo com a regulamentação da FEUP.

Bom trabalho!
RMA, JVV, HSF.

# A O módulo readline

- readline.h

```
/*
 * readline.h - readline command module
 */

#ifndef _READLINE_H_
#define _READLINE_H_

ssize_t readline(int fd, void *vptr, size_t maxlen);

#endif
```

- readline.c

```
/*
 * readline.c - readline command module
 */
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

#include "readline.h"

ssize_t readline(int fd, void *vptr, size_t maxlen) {
  ssize_t n, rc;
  char  c, *ptr;

  ptr = vptr;
  for (n = 1; n < maxlen; n++) {
    if ( (rc = read(fd, &c, 1)) == 1) {
      *ptr++ = c;
      if (c == '\n')
        break;
    } else if (rc == 0) {
      if (n == 1)
        return(0);  /* EOF, no data read */
      else
        break;     /* EOF, some data was read */
    } else
      return(-1); /* error */
  }

  *ptr = 0;
  return(n);
}

void example() {
```

```
    int fp;
36  int l;
    char buf[255];
38
    fp = open("pipe", O_RDONLY);
40  l = readline(fp, buf, 244);
    printf("read\n\t %s\n", buf);
42  close(fp);
}
```