MESTRADO INTEGRADO EM ENGENHARIA INFORMÁTICA
LICENCIATURA EM CIÊNCIAS DA COMPUTAÇÃO

Sistemas Operativos

Teste

2 de Junho de 2016 — Duração: 2h

I

1 No sistema de backup/restore do trabalho prático deste ano havia clientes, vários servidores, sinais e 1 FIFO. Que justificação tem esta aparente complexidade para algo que na sua essência consiste na cópia de ficheiros? Escolha dois objectivos desta abordagem, os que lhe parecerem mais importantes, e demonstre que sem esta arquitectura esses objectivos dificilmente seriam atingidos.[1]

2 Se tivesse de implementar um simples programa sequencial de cópia de ficheiros (ler de um lado, escrever noutro, não há servidores nem FIFOs), como faria o controlo de carga? Isto é, como garantia que no máximo estivessem em execução X operações de cópia? Note que este programa continua a poder receber muitos argumentos e a ser executado concorrentemente por outros scripts ou utilizadores.[2]

3 Em termos de CPU consumido, que tipo de carga coloca o serviço de backup/restore do trabalho prático? Para este caso concreto, qual a estratégia de escalonamento que aconselharia? Justifique (em poucas linhas).

II

Considere um programa `CTRL` que monitoriza a actividade de um conjunto de processos a executar no mesmo computador. Cada processo monitorizado escreve, de 3 em 3 segundos, "OK" para o seu standard output enquanto se encontra activo. O programa `CTRL` recebe como argumentos um número arbitrário de programas (`CTRL p1 p2 ... pn`) que deverá colocar em execução e monitorizar. Sempre que um processo deixe de estar activo (não for recebido "OK" durante 3 segundos) o processo deve ser terminado e o tempo aproximado em que esteve activo reportado para o standard output. Logo que todos os processos tenham sido terminados `CTRL` termina a sua execução.

III

Considere um sistema para geração de imagens "captcha" a pedido, onde um processo servidor gera imagens para potenciais clientes. Os clientes comunicam com o servidor por um pipe com nome, indicando uma palavra de 6 letras e a sua identificação. O servidor gera uma imagem para a palavra indicada e envia-a ao cliente respectivo, também via pipes com nome. Escreva uma função `create_captcha_file(const char* palavra)` para ser usada por programas cliente, que contacta o servidor e para uma `palavra xxxxxx` grava num ficheiro `xxxxxx.png` a imagem correspondente devolvida pelo servidor. Implemente também o programa servidor assumindo a existência de uma função `size_t captcha(const char* palavra, char * buffer)` que escreve no `buffer` a imagem correspondente a `palavra`, devolvendo quantos bytes forem escritos, no máximo 16384 bytes.

Algumas chamadas ao sistema relevantes

Processos

- pid_t fork(void);
- void exit(int status);
- pid_t wait(int *status);
- pid_t waitpid(pid_t pid, int *status, int options);
- WIFEXITED(status);
- WEXITSTATUS(status);
- int execlp(const char *file, const char *arg, ...);
- int execvp(const char *file, char *const argv[]);
- int execve(const char *file, char *const argv[], char *const envp[]);

Sistema de Ficheiros

- int open(const char *pathname, int flags, mode_t mode);
- int close(int fd);
- int read(int fd, void *buf, size_t count);
- int write(int fd, const void *buf, size_t count);
- long lseek(int fd, long offset, int whence);
- int access(const char *pathname, int amode);
- int pipe(int filedes[2]);
- int dup(int oldfd);
- int dup2(int oldfd, int newfd);

Sinais

- void (*signal(int signum, void (*handler)(int)))(int);
- int kill(pid_t pid, int signum);
- int alarm(int seconds);
- int pause(void);

---

[1] Pretende-se uma resposta sucinta, *menos de 10 linhas legíveis*.

[2] Para evitar ambiguidade de linguagem, responda em *código C*. Bastam 3 linhas...