Sistemes Operatius 1

Sessió de problemes – 3 de d'abril

Introducció

En aquesta sessió de problemes es presenten exercici relacionats amb la tercera pràctica. Aquesta sessió de problemes està centrada en fer servir canonades per comunicar dades entre processos així com els senyals que serveixen perquè processos s'enviïn "interrupcions" entre entre sí. Aquestes interrupcions, però, són un mecanisme gestionat a nivell de programari i no de maquinari.

Aquesta sessió no té cap exercici que s'hagi d'entregar i puntuï a la qualificació. Els problemes que s'hi plantegen són voluntaris. Els exercicis estan relacionats amb la pràctica a realitzar.

Les canonades

Les canonades és un mecanisme de comunicació entre processos que permet enviar informació d'un procés a un altre. Típicament es fa servir a la línia de comandes, per tal d'enviar la sortida estàndard d'un procés a l'entrada estàndard d'un altre procés.

Les canonades es faran servir aquí (i, en particular, a la pràctica 3) perquè un procés pugui enviar dades a un altre procés. Es faran diversos experiments perquè es pugui entendre bé els conceptes associats.

1. Començarem analitzant la mida de la canonada. Per això farem servir el fitxer `pipe_size.c`, el qual fa que el procés fill escrigui a la canonada. El pare, però, no llegeix cap cada de la canonada. Això farà que la canonada s'ompli. Observa que arriba un moment en què el programa és "penja". Per què és penja? Quina informació ens està donant el programa sobre la mida de la canonada?

2. A Linux existeix una forma d'augmentar la mida de la canonada. Teniu un exemple a `pipe_large.c`. Les crides a sistema que es fan servir per augmentar no formen part de l'estàndard POSIX i, per tant, no tenen perquè funcionar a altres sistemes Unix (com el Mac). Les podeu fer servir per fer la pràctica.

   Proveu el codi! Pot semblar útil tenir una canonada gran, però no és eficient, atès que una canonada gran fa que es facin servir més recursos del sistema operatiu. A més, existeixen altres sistemes més eficients de comunicació interprocés en cas que es vulgui transferir gran quantitat d'informació (pex comunicació via xarxa, fitxers, fitxers mapats a memòria). Tingueu en compte que la canonada és la forma més "antiga" de comunicació interprocés.

3. A continuació farem que el buffer de la canonada s'ompli pel fill. Després el pare llegirà les dades escrites pel fill. Teniu un exemple al codi `pipe_write_read.c`. Analitzeu el codi, executeu i observeu què fa. Observeu que el fill acaba abans que el pare comença a llegir! A més, veureu que el codi es "penja" al final. Per què?

4. Finalment, farem una prova amb un codi que genera un "flux continu" de dades: el fill hi escriu dades i el pare les llegeix. Teniu el codi al fitxer `pipe_write_read_continuous.c`. Aquest darrer exemple és la forma en què es comuniquen dos processos quan generem una canonada.

Exercicis

Es proposen a continuació un parell d'exercicis, relacionats amb la pràctica, que fan servir les canonades.

1. Al fitxer pipe_size.c hem vist la mida, en bytes, del buffer associat a la canonada. Modifiqueu el codi per escriure-hi valors sencers. Aneu incrementant el valor que hi escriviu al buffer. Quants valors sencers hi podeu escriure?

2. Agafeu el fitxer `pipe_write_read.c` perquè el fill hi escrigui sencers en format binari (i.e. fent servir les funcions read i write). Aneu incrementant el valor que hi escriviu al buffer. Un cop el buffer sigui ple, el pare començarà a llegir els valors. Imprimiu els valors llegits per pantalla per assegurar que els valors que llegiu són els correctes.

Els senyals

Els senyals són un mecanisme de comunicació molt senzill entre processos. Són interrupcions de programari que dos processos poden enviar entre sí per senyalitzar que ha tingut lloc un esdeveniment. Vegem-ne un parell d'exemples

1. El fitxer `sigterm_sigint.c` conté un exemple en què l'aplicació espera a rebre un senyal. Observeu el codi i indiqueu quin és el comportament d'aquest: quan imprimirà el missatge "Exiting main"? Quan imprimirà el missatge "Emergency exit"? Com ho aconsegueixes fer?

2. El fitxer `sigprocesses.c` conté un segon exemple en què dos processos, pare i fill, s'envien senyals entre sí. Aquest és un mecanisme de sincronització senzill entre dos processos (a l'actualitat hi ha mecanismes de sincronització més avançats que es veuran a Sistemes Operatius 2 com, per exemple, els semàfors). Com sap el pare quin és el seu fill? Com sap el fill quin és el seu pare?