Stor övning på kommunikation mellan processer

Övningen har tre uppgifter, de första två är bra att göra, den sista uppgiften är för de som är ute efter något extra att bita i. Först ska du skriva ett *C*-program som utför tre *UNIX*-kommandon med hjälp av tre olika barnprocesser och kopplar ihop barnen via pipar. Sedan, i uppgift 2, ska du byta ut en av piparna mot en *UNIX*-domain socket och köra övningen i två olika program som då kommunicerar med IPC över en socket. När vi går in i kursens tredje del kan även detta göras över IP mellan två datorer och då behövs alltså en Internet-domain socket. Vi kommer att gå igenom *UNIX*-domain sockets på kursmöte 5 och Internet-domain sockets tillhör nästa kursdel.

Uppgift 1 a) Det som ska göras uttrycks i BASH så här:

```
> ls -l /bin/?? | grep rwxr-xr-x | sort
```
(Testa gärna vad detta gör.)

I BASH skapas tre barnprocesser till skalet som är en föräldraprocess. Nu ska du göra detta i *C*. Barn 1 ska utföra `ls -l /bin/??` och skicka resultatet vidare via pipe 1 till barn 2 som ska utföra `grep rwxr-xr-x` osv. Det kan behövas flera pipar. Alla pipar ska stängas ordentligt och alla barnprocesser ska inväntas ordentligt så att inga zombier uppkommer.

Ledning: Det kommer inte att gå att skriva kommandot "`ls -l /bin/??`" direkt in i en `execlp()` som man kanske vill från början. Anledningen är att jokrarna (`??`) förutsätter en så kallad skalmiljö (environment). Ett sätt att åstadkomma denna skalmiljö är att köra kommandot "`ls -l /bin/??`" i ett mindre skal (`sh`) med `execlp()`, i *C* kommer det att se ut ungefär så här: `execlp("/bin/sh", ..., "ls -l /bin/??", ...);`. Du behöver studera manualsidor och experimentera lite grann för att veta exakt hur anropet ska se ut. Testa `man bash` och `man sh`. Detta är också nyckeln till laboration 1 – hur man får ett skal att köra ett kommando baserat på en textsträng.

Uppgift 1b) Rita ett tidsdiagram för de olika processer som skapas, diagrammet ska även illustrera de ingående processernas relationer, vem som kommunicerar med vem etc, vem som uför vilket kommando etc.

Uppgift 2. Byt en av piparna mot en *UNIX*-domain socket och skriv om labben så att det är två olika program som kör. (Som sagt, vi går igenom sockets nästa kursmöte.)

Uppgift 3. Den här uppgiften kräver att du sätter dig in en del i vad en *partition* är för något. Använd kommandona `dd`, `mkfifo`, `bzip2` och `bunzip2` för att skapa en fil- och partitionskompressor. Skriv den först i bash och basera dig på följande ide om hur man kan komprimera en hel partition (här `/dev/sda11` på 512 MB):

```
bzip2 < pipe > sda11.bz2 &
dd if=/dev/sda11 of=pipe count=1968284 bs=1024 &
```

Använd signaler till `dd`-processen för att kunna skriva ut en löpande procentsats gällande hur mycket av partitionen som har komprimerats. (`man kill`, `man dd`). Skriv sedan samma program i *C* med pipes enligt den kommunikationsmodell vi gått igenom ovan. Lägg även till uppackning. Detta är ganska kompakt formulerat, en mer tillgänglig formulering finns i `extrauppgift.pdf`. Där finns också bättre referenser till `coreutils` etc. Kolla den.