Lektion 2 - Introduktion till systemnära funktioner.

Denna lektion introducerar några för obligatorisk uppgift 1 viktiga funktioner. Målet med lektionen är att hjälpa dig i ditt lärande av några nya användbara funktioner. Du är fri att välja ifall du vill jobba ensam, med andra kursdeltagare eller att inte göra uppgifterna.

0. Läs igenom några manualsidor genom kommandot `man`. Några exempel är `man string`, `man 3 exec`, `man 3 fork`, `man man`. 3:an krävs för att säkerställa att du öppnar en manualsida för ett C-program och inte för ett program/kommando/systemanrop med samma namn.

1. Skriv ett kort program som skriver ut texten "Testing" samt alla programargument till `stdout`. Kompilera källkoden och döp den exekverbara filen till *test_prog*. Verifiera att du kan köra programmet.

2. Skapa ett nytt program med namnet *exec.c*. Detta program ska med någon utav *exec*-funktionerna köra ditt program. Kom ihåg att kolla returvärdet på funktionen och skriv ut ifall något går fel. Kolla upp funktionen `perror` i manualsidorna och kom ihåg 3:an...

3. Lek runt lite med olika versioner av *exec*, testa vad som händer när du skriver in fel saker osv. Testa även ifall allt fungerar som det ska ifall du kör programmet med ett inbyggt program som `ls` istället för *test_prog*. Använd `man ls` för att se vad kommandot `ls` gör.

4. Skriv ett program, *fork.c*, som använder funktionen `fork`, sedan `printf("Hello\n")` och till sist avslutar.

5. Uppdatera programmet så att bara föräldraprocessen skriver ut sitt barns processid.

6. Uppdatera programmet så att barnet exekverar *test_prog*.

7. Lägg till funktionen `wait` i föräldern, samt en `printf("Bye\n")` efter den. Uppdatera även *test_prog* med funktionen `sleep` innan och efter utskriften.

8. Skapa ett nytt program med namnet *pipe.c*. Detta program ska skapa en *pipe* med hjälp av funktionen `pipe`. Sedan ska programmet först skriva till *pipe*:n, därefter läsa från den och skriva ut resultatet. De enklaste funktionerna som du kan använda för att läsa och skriva till *pipe*:n är `write` och `read`.

9. Uppdatera programmet så att skrivandet till *pipe*:n sker i en process och läsandet i *pipe*:n sker i en annan process (använd funktionerna `fork` och `wait`).

10. Läs på om vad funktionerna `dup` och `dup2` gör. Skapa ett nytt program med namnet *dup.c* som öppnar en fil i *write*-mode. Du ska sedan använda

`dup2` för att koppla `stdout` direkt till filen, så att det du skriver ut med `printf` hamnar direkt i filen. `stdout`, `stdin` och `stderr` är vanliga fildeskriptorer som initialt är öppna och används för att skicka information genom terminalen via bland annat printf och scanf.

11. Skriv en funktion `char **parse_line(char *buffer)` som tar en sträng med ett kommando (kommando plus eventuella argument) och skapar en array av pekare till null-terminerade strängar som innehåller varje del av kommandot. Arrayen av pekare till strängar ska avslutas med en pekare till `NULL`. Den skapade arrayen ska kunna användas som argument till `execvp`. Skriv ett program för att testa funktionen. Tips: Läs *man*-sidorna för funktionerna `strdup` och `strtok`.

Bra jobbat!