INTRODUKTION TIL UNIX®

20. november 1984

Hvad er UNIX® og hvorfor er det blevet så populært.

UNIX er et operativsystem som i den seneste tid er blevet mere og mere populært på mange 16- og 32-bit minicomputere. I modsætning til de mange enkeltbrugersystemer (f.eks. CP/M og MSDOS) som idag tilbydes til mini- og microcomputere, er UNIX et avanceret flerbrugersystem med mange spændende faciliteter.

UNIX Historie

En viden om UNIX historie er vigtig for forståelsen af, hvorfor UNIX operativsystemet er, som det er. UNIX har en ret lang historie bag sig og mange mennesker har været beskæftiget med systemet. UNIX har aldrig været defineret som et projekt, men alle som har arbejdet med systemet har tilføjet de faciliteter som han/hun har manglet. De bedste programsystemer er efterhånden blevet end del af det man måske kunne kalde "standard UNIX". Denne måde hvorpå UNIX er blevet til, forklarer hvorfor operativsystemet kan beskrives som "et system for programmører lavet af programmører".

UNIX opstod på Bell Laboratories i USA i slutningen af 60´erne, fordi Bell trak sig ud af et samarbejde omkring Multics timeshare systemet på Massachussetts Institute of Technology. En af medarbejderne på Bell Laboratories, Ken Thompson, stod nu pludselig uden datamaskine til at videreføre sit projekt, men fandt snart en PDP-7 datamaskine, hvor programudviklingen var baseret på hulstrimler. Ken Thompson savnede det interaktive programudviklingssystem fra Multics, og begyndte derfor at skrive et enkeltbrugeroperativsystem, som han kaldte UNIX, i modsætning til Multics. Mange af ideerne i UNIX kan føres tilbage til Multics.

UNIX blev kendt af mange på Bell Laboratories bl.a. Dennis Ritchie som hjalp Ken Thompson med at flytte UNIX til en PDP-11, ændre det til et multibrugersystem og udvikle forskellige hjælpeprogrammer.

I begyndelsen af 70´erne var der stor interesse for flytbare programmer og assemblerlignende sprog med strukturer fra bl.a. Algol. Et sådant sprog var f.eks. BCPL og det havde vist sig at programmer skrevet i BCPL var meget lette at flytte til andre maskiner. Dennis Ritchie udviklede på baggrund af BCPL et nyt programmeringssprog, som han kaldte C, specielt beregnet til flytbare programmer og i 1973 var UNIX skrevet om til C.

Siden er UNIX blevet flyttet til mange maskiner og mange firmaer sælger nu UNIX på licens fra Bell til deres maskiner.

UNIX findes i dag i mange udgaver og versioner. En af de tidligst udbredte versioner var den såkaldte version 7 som fremkom i slutningen af halvfjerdserne.

Denne version lever stadig som basis for flere kommercielle udgaver f.eks. XENIX som er en meget udbredt UNIX version fra firmaet Microsoft. Senere kom en version, som blev kaldt system III og som er basis for mange nyere UNIX-systemer f.eks. PC/IX som er IBM´s udgave beregnet til deres personlige computere, HP-UX som Hewlett-Packard leverer til deres superminier.

De sidste udgave som er kommet er SV som nu også markedsføres af AT&T, som er Bell Laboratories moderfirma.

En speciel status har Berkeley Universitetet i Californien haft. De har udviklet en række UNIX versioner, som dog ikke har været kommercielt tilgængelige men som på mange måder har været forbedret i forhold til de verioner Bell Laboratories har udviklet.

Versionerne fra Berkeley er meget udbredt på universiteterne og har navne som bsd4.2 (Berkeley system distribution 4.2).

Mange kommercielt tilgængelige systemer benytter meget fra Berkeley udgaverne f.eks. skærmeditoren vi og kommandofortolkeren csh.

UNIX egenskaber

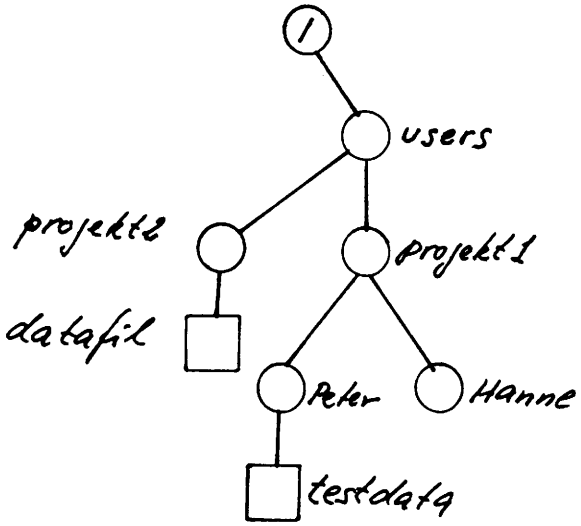
UNIX har nogle egenskaber som især har bidraget til dets succes og som vil blive beskrevet i det følgende.

Katalog og filstruktur

UNIX har en såkaldt hierarkisk filstruktur. Filsystemet består af et hovedkatalog kaldet ROOT (med betegnelsen /). I dette katalog kan der dels optræde datafiler, dels kataloger. I katalogerne kan der igen optræde enten datafiler eller kataloger og dette kan gentages i mange niveauer.

Det at oprette et katalog er i UNIX ikke nogen omstændelig ting. Enhver bruger kan skabe et nyt katalog når han/hun synes, at der er behov for det.

Når en fil skal refereres kan man enten angive hele vejen eller stien fra rodkataloget til den pågældende fil. Der startes med rodkataloget og derefter opgives de kataloger der skal gennemsøges for at finde den ønskede fil f.eks. /users/projekt1/Peter/testdata (se figur).



Et bestemt katalog kan udpeges om arbejdskatalog. Er et sådant katalog blevet udpeget skal kun vejen fra dette katalog til filen beskrives. Sættes arbejdskataloget f.eks. til /users/projektl/Peter kan filen testdata refereres kun med brug af filnavnet.

Enhver bruger har et på forhånd udpeget katalog som arbejdskatalog når han/hun logger på systemet. Dette katalog kaldes loginkataloget.

Som yderligere hjælp ved arbejdet med kataloger findes notationen . for det øjeblikkelige arbejdskatalog, og .. for det katalog som arbejdskataloget er registreret i. Er f.eks. arbejdskataloget projektl, og man et øjeblik ønsker at benytte en fil, f.eks. datafil, i kataloget projekt2, kan denne fil refereres som ../projekt2/datafil. Dette forudsætter dog at man har tilladelse til at søge i kataloget projekt 2.

Filer i UNIX har en meget simpel struktur; der er nemlig ingen struktur. En fil opfattes som en samling bytes og adderes en byte til en file bliver denne fil en byte længere! Alle bytes kan læses eller skrives enkeltvis, f.eks. kan man fra et program bede om byte nr. 1736 i en fil (hvis der er så mange!). Der er altså i forbindelse med filen ikke registreret oplysninger om postlængde, bloklængde, blokningsfaktor etc.

Dette betyder ikke at filer ikke kan læses og skrives som "direct access" filer med fast postlængde, men en sådan læsning eller skrivning defineres af det program, som benytter filen; det er ikke registreret på filen. Dette betyder at en fil f.eks. kan skrives med en recordlængde på 100 bytes og læses med en recordlængde på 80 bytes.

Sikkerhed

Når en person logger på et UNIX-system benyttes et brugernavn som systemet skal kende på forhånd, evt. skal også et password opgives. Brugerne kan samles i forskellige grupper således at systemet kender en bestemt bruger ved navn og gruppe, f.eks. Hanne i gruppen af programmører.

Når en fil skabes, noteres det i kataloget, hvem der ejer filen (normalt den person som skaber den) og hvem der har adgang til filen. Adgangsforhold til filer kan specificeres i 3 niveauer: adgangsforhold for ejeren, adgangsforhold for personer i gruppen og adgangsforhold for andre. For hvert af disse niveauer kan det specificeres om en person har lov til at læse filen, skrive i filen, udføre filen (hvis den indeholder et program).

Eksempelvis vil det være muligt for programmøren Hanne at oprette en fil som hun kan læse, skrive og udføre fordi hun ejer filen. Alle programmører kan læse og udføre programmet i filen fordi de er i gruppe med Hanne og andre kan kun udføre programmet.

Shell

UNIX shell er på en gang et kommandosprog og et programmeringssprog. Når man logger ind på et UNIX system startes automatisk et på forhånd udpeget program. Dette kan enten være et anvendelsesprogram eller en kommandofortolker f.eks. shell. Shell udfører kommandoer som enten læses fra en terminal eller fra en fil. Den grundlæggende kommandostruktur har følgende form:

navn -options argumenter.

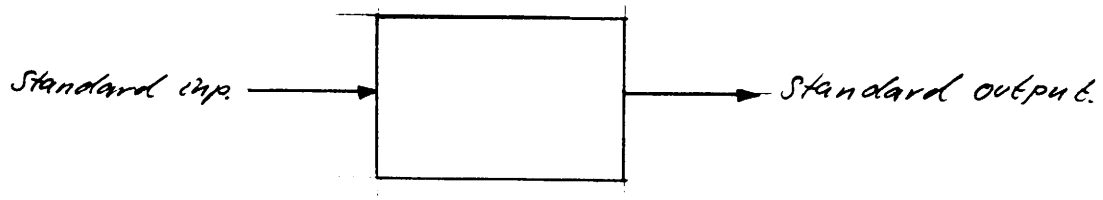
Navn er navnet på kommandoen f.eks. sort; options angiver forskellige muligheder for at udføre kommandoen f.eks. sort -r, sorterer i faldende orden. Argumenterne angiver hvad kommandoen skal udføres på, f.eks. hvilke filer der skal sorteres.

Kommandoer er normalt udført som selvstændige programmer og for at udføre kommandoen skaber shell en ny process og venter på at denne bliver færdig.

Shell kan også udføre en kommando og undlade at vente på, at denne bliver færdig. Dette sker ved at tilføje operatoren & til slutningen af kommandolinien. F.eks. vil print file & starte en udskrivning af filer i baggrunden, mens shell fortsætter. & er en metakarakter, d.v.s. en karakter, som har en særlig betydning for shell, og tegnet vil derfor ikke blive overført til kommandoen som argument.

Dynamisk tildeling af I/O

UNIX opfatter et program som værende en transformation af data, d.v.s. en strøm af inddata som transformeres af programmet til en strøm af uddata.



Strømmen af inddata kaldes for "standard input", og strømmen af uddata kaldes for "standard output".

Standard input, som normalt er defineret som værende tastaturet, er således programmets vigtigste input, og hovedparten af output produceres på standard output, der normalt er defineret som værende skærmen.

Standard output for en kommando kan dirigeres til en anden enhed eller en fil ved hjælp af operatoren >, f.eks. ls>file. Kommandoen ls (skriv indholdet af et katalog) skrives nu på filen file i stedet for på skærmen. Hvis filen ikke findes bliver den skabt, hvis den findes skrives der forfra på filen. Der findes en notation for at tilføje til en fil, nemlig >>, ls>> file vil således tilføje en katalogudskrift til filen file.

Tilsvarende kan en kommando læse sine data fra en fil i stedet for fra tastaturet.

Kommandoen wc vil læse en fil, og når endfile er læst vil den skrive antallet af karakterer, ord og linier på standard output. Ønskes denne kommando nu anvendt på filen testfile skrives wc<testfile. Operatoren < anvendes altså til at angive hvilken fil, som er standard input.

Pipes og Filtre

En væsentlig egenskab i shell er muligheden for direkte at forbinde standard output fra en kommando til standard input i en anden. En sådan forbindelse kaldes en "pipe" og symbolet | bruges til at danne forbindelsen. ls|wc skriver således antallet af filer i et katalog.

De to processer ls og wc er synkroniserede, således at wc venter, hvis der ikke er data nok og ls venter hvis wc ikke kan behandle data så hurtigt som ls producerer dem.

Mange programmer i UNIX har som vigtigste opgave at transformere data i forbindelse med en "pipe". Sådanne programmer kaldes ofte for filtre. Et sådant filter er f.eks. grep, som fra standard input kun sender de linier hvori et givet mønster findes. grep old < testfile skriver fra testfile de linier hvori strengen old findes. En pipeline kan bestå af flere kommandoer f.eks. vil:

```
ls|grep old|wc
```

skrive antallet af linier fra katalogudskriften hvori ordet old forkommer.

Generering af filnavn

Mange kommandoer accepterer filnavne som argumenter. F.eks. vil ls -l prog.c skrive katalog information om programmet prog.c. Shell har indbygget en mekanisme, som udfører en kommando med alle filnavne, som passer til et givet mønster. ls -l *.c skriver kataloginformation om alle programmer, som ender på .c, d.v.s. alle c programmer.

* er et mønster, som passer til alle mønstre af vilkårlig længde (inkl. 0 længde). Tilsvarende passer ? til enhver enkeltkarakter og abc til enten a, b eller c. a-d svarer til enhver karakter i området fra a til d. Disse metakarakterer kan kombineres til mere komplicerede mønstre, f.eks. vil mønstret a-z * passe til alle mønstre som begynder med en karakter i området a-z og derefter en vilkårlig streng.

Shell Procedures

Shell kan udføre en række kommandoer, som først er skrevet i en fil.

sh file arg1 arg2.... vil udføre programmet sh (shell), altså en ny udgave af kommandofortolkeren, og denne udgave vil læse sine instruktioner fra filen file.

En fil med kommandoer til shell kaldes et shell script.

Argumenterne arg1, arg2 .... som gives ved kaldet, kan man i scriptet referere til som $1, $2 ... Et shell script kan således generaliseres og arbejde på forskellige aktuelle argumenter, som gives ved kaldet, analogt til subrutiner eller procedurer i programmeringssprog.

Ex: Shellscriptet antal indeholder kommandoen

```
ls|grep $1|wc
```

```
sh antal old
```

vil være ækvivalent til kommandoen

```
ls|grep old|wc
```

Afslutning

Der kan nævnes mange årsager til at UNIX er blevet så populært. En af dem er naturligvis den lette flytbarhed af programmer der er skrevet til UNIX fra en type maskine til en anden. Specielt er også operativsystemet let at flytte fra en maskine arkitektur til en anden.

En anden årsag er det meget store udvalg af kommandoer og nytteprogrammer, der som standard følger med ethvert UNIX system.

De fleste af disse kommandoer er meget fleksible og da de ydermere let lader sig kombinere ved hjælp af pipes, giver det mulighed for at løse mange mindre opgaver ved brug af allerede eksisterende programmer og kommandoer i stedet for at skriv et specielt program. Under anvendelse af de mange faciliteter i shell kan også ret komplicerede problemer løses og mange programmører hævder at deres produktivitet er forøget væsentligt under anvendelse af UNIX.

Om UNIX, som nogen hævder, vil løse den såkaldte softwarekrise er nok tvivlsomt men der er ingen tvivl om et UNIX er kommet for at blive og at vi vil høre mere til det i fremtiden.