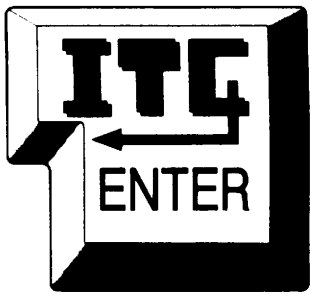
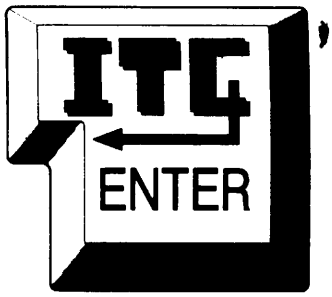


Dansk POSIX guide

Første udgave december 1988

Informations Teknologi Center
Ny Adelgade 5A, 2sal
1104 København K

Dansk Standardiseringsråd
Aurehøjvej 12
2900 Hellerup



Dansk POSIX guide

baseret på de standardiseringstiltag som finder sted i

ISO/IEC JTC 1 SC 22/WG 15

og som i Danmark varetages af

arbejdsgruppe A1 (POSIX og "C")

underudvalg u22 (Programmeringssprog og Operativsystemer)

hovedudvalg S142 (Informationsteknologi),

under Dansk Standardiseringsråd.

Denne guide er udarbejdet og produceret af Isak Korn fra Informations Teknologi Center. Arbejdet med udarbejdelsen og udgivelsen af denne guide er financieret og tilset af Dansk Standardiseringsråd.

Guiden er baseret på materialer som er blevet tilgængelige i forbindelse med aktiv deltagelse i det internationale standardiseringsarbejde som er blevet initieret af International Standardiserings Organisation.

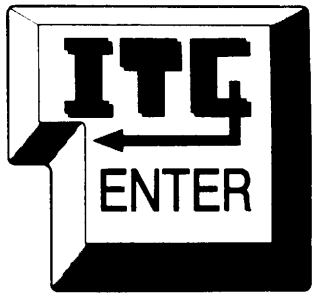
Første udgave december 1988.

Dokumentet er sat og formateret med brug af FrameMaker-systemet, der kører under UNIX på SUN Workstation hos ETC, København.

Informations Teknologi Center
Ny Adelgade 5A, 2sal
1104 København K

Dansk Standardiseringsråd
Aurehøjvej 12
2900 Hellerup

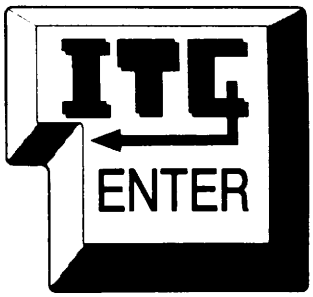
INDHOLDSFORTEGNELSE



Dansk POSIX guide

Del 1

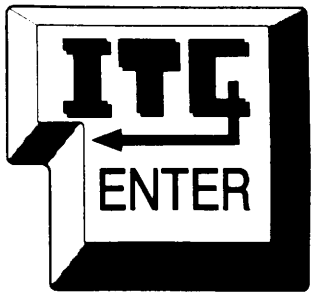
Lidt historie omkring standarden



Tilblivelse af en standard i ISO's regi

Selve arbejdet med standardiseringen af et område på international plan varetages af ISO eller IEC. Med hensyn til informationsteknologi området har ISO og IEC dannet en fælles teknisk komité som varetager standardisering indenfor de relevante emner. Denne komité betegnes som JTC 1 og er opdelt i 24 subkomitéer som varetager standardiseringen af de forkellige emner som er relaterede til IT området. En af disse subkomitéer er SC 22, som bekæftiger sig med programmeringssprog og operativ systemer samt de miljøer som der kræves af både programmeringssprog og operativ systemer.

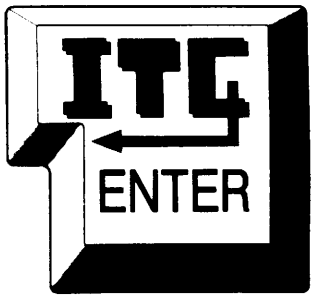
Det første step med hensyn til standardiseringen af et emne består i at en nation eller en af subkomitéerne foreslår dette emne som såkaldt forslag til et nyt arbejdsområde med hensyn til at fremstille en standard. Dette kaldes PNWI eller NWI (Proposed eller New Working Item). Dette forslag bliver sendt til alle medlemsnationer af ISO for en afstemning af 3 måneders varighed. Hvis emnet opnår tilstrækkelig support fra medlemsnationerne, bliver dette ophøjet til et projekt som "kører" både i international og national regi med dannelse af de internationale og nationale arbejdsgrupper til følge. Når PNWI eller NWI sendes til afstemning angiver forslagstilleren hvilket dokument man har tænkt sig at anvende som såkaldt "arbejdsdokument" (WD). Med udgangspunkt i arbejdsdokumentet bearbejder arbejdsgrupperne forslaget til den standard som de er igang med at definere. På baggrund af denne procedure kan det oprindelige dokument ændre sit udseende meget i løbet af den tid arbejdsgrupperne beskæftiger sig med dokumentet. Når arbejdsgrupperne er nået til enighed om at dokumentet er blevet stabilt både teknisk og indholdsmæssigt så kan dette dokument registreres som en "Draft Proposal" (DP) og blive sendt til afstemning og kommentarer blandt medlemmer af ISO. Denne periode er berammet til at være 3 måneder.



Derefter bearbejder arbejdsgruppen de eventuelle kommentarer som måtte komme som følge af afstemning/kommentar runden. Hvis der under kommentar runden ikke har været mange tekniske kommentarer til det foreslåede, så kan dokumentet efter at have været rettet til blive registreret som "Draft International Standard" (DIS) og sendt til en ny afstemning, denne gang i en periode på 6 måneder. Ellers skal de tekniske kommentarer besvares eller indkorporeres i dokumentet og så skal det igen sendes til en afstemning som en DP for 3 måneder. Når dokumentet bliver registreret som DIS og den efterfølgende afstemning viser tilslutning uden kommentarer af teknisk art, editoriale kommentarer er tilladte, så kan den nye standard publiceres som en international standard – IS.

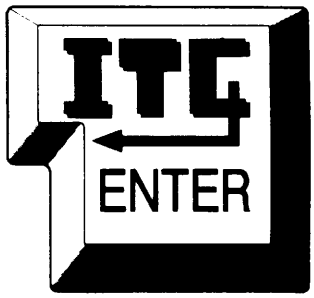
Med hensyn til POSIX standarden så er den første del af standarden blevet registreret som DIS og udsendt til afstemning blandt medlemsnationer. Man kan nævne at i den korte tid ISO har beskæftiget sig med POSIX, har vi oplevet fem modificerede dokumenter som blev bearbejdet både i Danmark og internationalt. Da vi startede på POSIX arbejde i ISO's regi i september 1987 var den gældende version af dokumentet "Draft 7", mens den registrerede version bærer nummer 13 ("Draft 13").

Lidt mere generelt om tilblivelsen af standarder, kan læses i Tillæg 3 til denne Guide.



Indledning og et historisk rids

POSIX (Portable Operating System Interface), er i sit nuværende udseende baseret på UNIX(TM) operativ systemet, derfor vil det være på sin plads at starte med lidt historie omkring netop UNIX. UNIX operativ system er et multiprocess (eller multitask) system, hvis faciliteter funktionsmæssigt ligner andre minicomputer eller mainframe operativ systemer. Men UNIX systemet er også et fænomen. Det er udviklet af en lille gruppe mennesker, egentlig til eget brug, men må idag anses for at være det måske mest udbredte operativ system for såkaldt "mid-range" af komputere. Dette vel og mærke stort set uden at have gjort en indsats for at markedsføre systemet i den traditionelle forstand. Når UNIX således er blevet skubbet i forreste række er dette ikke sket via en fælles kampagne, men ved hjælp en tilbundsgående entusiasme hos systemets brugere. Disse brugere har allesammen forskellige opfattelser af og meninger om systemet, hvilket så igen medfører, at UNIX's image fremstår noget uklar. Dette kapitel har til hensigt, at gøre billedet klarere ved at belyse nogle spørgsmål som måtte gøre sig gældende når man har defineret UNIX som udgangspunkt for arbejde med standardiseringen af POSIX.



Årsagerne til systemets styrke

Det er tre af systemets grundlæggende egenskaber, der skal have æren for UNIX's store popularitet:

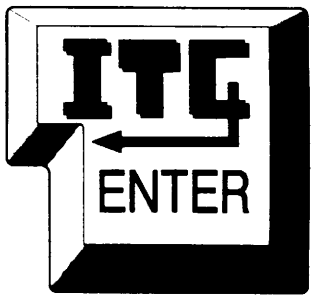
1. UNIX er et bærbart operativ system. Hardware forhandlere er interesserede i det fordi de relativt let kan implementere systemet på en ny komputer. På denne måde får de en lang række brugere. Eftersom systemet kan implementeres på så mange forskellige maskiner, stilles både software leverandører og brugere temmelig frit med hensyn til valg af hardware når de anskaffer et UNIX baseret system. Et program som er udviklet i UNIX miljø kan anvende operativ systemets faciliteter direkte og stadig blive overført til en anden maskine blot ved genoversættelse. Men det er ikke kun programmerne, der let flyttes fra en UNIX maskine til en anden, det samme gælder for værktøjerne og operatørernes ekspertise. Medarbejdere, der er oplært på en UNIX implementering, kan næsten lige med det samme bruge deres viden på en anden UNIX maskine.

2. UNIX som operativ system danner basis for et yderst produktivt databehandlingsmiljø, især for programmørerne. UNIX indeholder en lang række værktøjer til udvikling af software, hvoraf en hel del kan anvendes direkte i "almindelig" tekstbehandling (dokumentation for eksempel). Endvidere kan disse værktøjer kombineres og anvendes til at løse et betragteligt antal problemer uden programmering. Kort sagt, supporterer UNIX sandsynligvis forbilledet for genanvendelig software bedre end noget andet operativ system på markedet.

3. Sidst, men ikke mindst er UNIX operativ systemet elegant i sit design, hvilket gør det udfordrende og tilfredsstillende at bruge, mange erfarne brugere vil endog kalde det morsomt at anvende. Det kan være vanskeligt at opsnappe essensen i systemets elegance, men det

hænger igen sammen med den styrke, der ligger i det enkle snarere end i det indviklede. Istedet for den "smørrebrødsseddel" over halv-uafhængige (og til tider sammenfaldende) egenskaber, der kendetegner mange operativ systemer, er UNIX udrustet med et rimeligt begrænset, men dog universelt antal faciliteter. Det får større bredde ved at fjerne begrænsningerne end ved at tilføje nye optioner. En af systemets designere udtrykker det således: "Det er et simpelt, sammenhængende system, der til fulde udnytter nogle enkelte gode ideer og modeller" (Dennis M. Ritchie, "Reflections on Software Research"). Samtidigt kan brugerne kombinere disse simple mekanismer og således skabe nye stærke redskaber, der er i stand til at imødekomme deres specielle ønsker og krav.

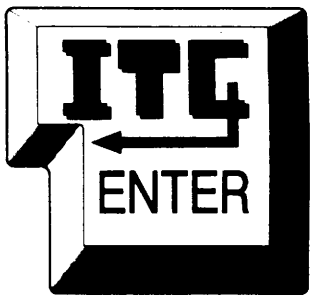
Det er lige til at forstå og påskønne værdien af et bærbart operativ system. UNIX's egenskaber med hensyn til elegance og forbedret produktivitet er mindre håndgribelige. Denne guide vil forsøge at belyse disse egenskaber, men en tilbundsgående forståelse for dem kan kun opnås, når man har arbejdet med systemet i nogle få måneder. Ikke desto mindre er det disse egenskaber, der nærer UNIX systemets bruger entusiasme. Det vil være yderst vanskeligt at finde en bruger, der forstår UNIX til bunds og som frivilligt ville vælge et andet operativ system på markedet. Systemets stærke sider opvejer helt klart de mangler, der vil blive beskrevet i det følgende afsnit. Ligeledes har UNIX's fleksibilitet været hovedårsag til at standardiseringen af et portabelt interface mellem operativ systemet er blevet baseret på UNIX, men på lang sigt skal POSIX (interface) være uafhængig af UNIX.



De svage punkter

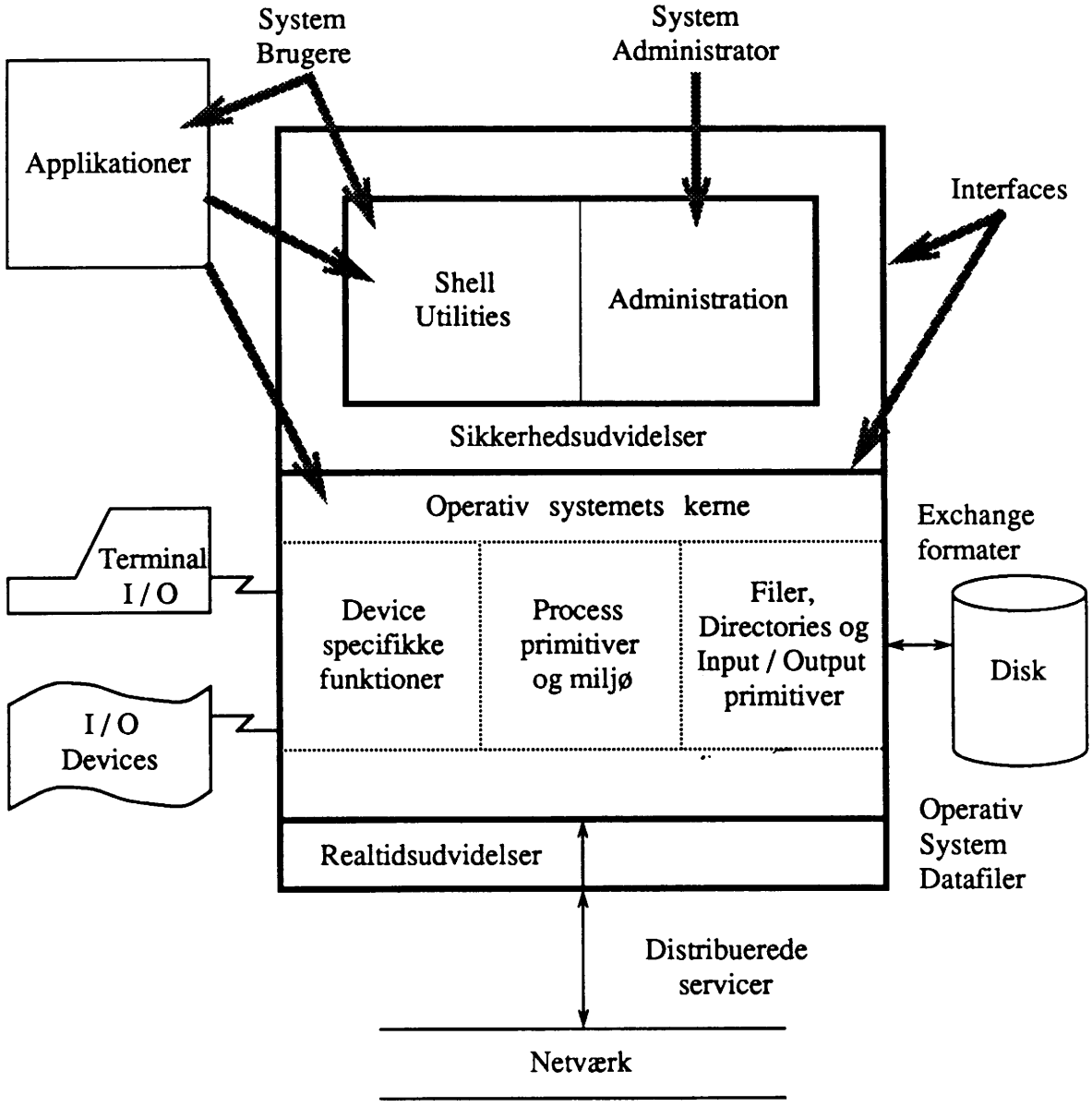
Kritikerne af UNIX fokuserer almindeligvis på den brugerflade som operativ systemet har for nye brugere (Det er dog rimeligt her at pointere, at mange avancerede brugere er fortalere for systemets brugerflade, specielt når det drejer sig om dets klare opbygning og dets underforståede respekt for deres kompetance.) Kommandofladen (shell) er kortfattet og utilstrækkelig, fejl rapporteringer er ikke gennemarbejdede og systemet har ingen "help" faciliteter. Nogle kommandoer har underlige navne som `grep` og `awk`. Systemet forudsætter, at brugerne ved hvad de laver for eksempel dets standard (default) løsning med at overskrive allerede eksisterende filer når en ny fil med samme navn oprettes.

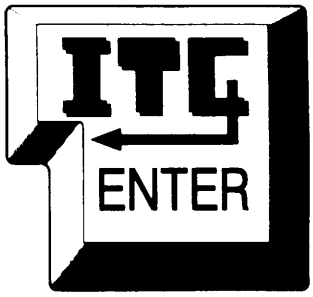
Hvorfor forekommer sådanne svage punkter overhovedet i et så populært system? De er resultatet af det miljø UNIX er blevet udviklet og anvendt i fra starten af. Dette miljø var Bell Laboratoriernes computer forskningsgruppe i de tidlige 1970ere. Medlemmerne i denne gruppe var databehandlingseksperter, og de konstruerede systemet til eget brug, de havde ingen planer om at udvikle et kommercielt operativ system. Det hardware de havde til rådighed bestod af en lille minicomputer udstyret med langsomme terminaler af Teletype typen. De udformede systemet således, at operativ systemets hjælpeværktøjer, der udgør en så stor del af brugerfladen, kunne udbygges af brugerne selv, hvilket også var tilfældet. Det er ikke overraskende, at der i dette miljø opstod et enkelt og tidsbesparende system, men med en utilstrækkeligt brugerflade (husk de langsomme terminaler med deres ekspert brugere). UNIX bredte sig til at begynde med til lignende miljøer, nemlig forkningslaboratorier og til univeriteternes computer-forsknings afdelinger. Brugerfladen udgjorde ikke nogen egentlig vanskelighed for disse grupper, specielt ikke når man tog systemets stærke sider i betragtning. Heller ikke da professionelle programmører begyndte at anvende systemet ude omkring i de enkelte



virksomheder, udgjorde brugerfladen et større problem. Nu forholder det sig imidlertid således, at systemet anvendes i flere forskellige miljøer. Brugerne af systemet dækker hele spektret, lige fra folk med ringe kendskab til databehandling til eksperter. Som følge heraf får de vanskeligheder nybegynderne støder på i forbindelse med brugerfladen tilsvarende større opmærksomhed. Der kan også være en anden årsag til, at man har været så længe om at ændre brugerfladen; hardwaren som var nødvendig til for at effektivt forbedre brugerfladen er blevet tilgængeligt kun i det sidste stykke tid.

Operativ system miljø
(Operating System Environment)
baseret på UNIX modellen

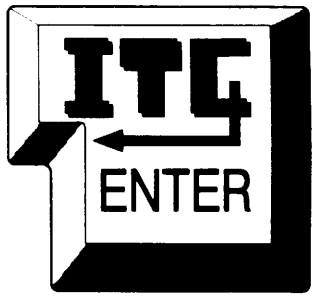




Evolutionen fra UNIX til POSIX

Brugerne af UNIX systemer har ændret sig siden systemet oprideligt blev udviklet, ligesom de problemer som UNIX er i stand til at løse, har ændret sig løbende. Men den hardware ved hjælp af hvilket UNIX fungerer har undergået en endnu mere drastisk ændring. Mikrocomputere med multimegabyte adresse felter er blevet almindelige, og hurtige skærmterminaler har fuldstændig erstattet Teletype skrivere. I stadig større udstrækning må tidsdelingssystemer (timesharing), vige for hele netværk af speciel indrettede arbejdsstationer. For at holde trit med de stadig skiftende modeller for anvendelse og udvikling af hardwaren, har UNIX undergået en række forandringer i løbet af årene, ligesom det vil være tilfældet i fremtiden. Samtidig har systemet været nødsaget til at være det samme og forblive med at være det samme for at bibeholde en af de vigtigste stærke sider – flytbarheden (portabiliteten). Disse krav, som kan forekomme paradoksale, kan dog forenes ved omhyggeligt at skelne mellem snitflader (interfaces) og implementeringer.

Der findes to former for snitflader, der kan være af interesse, programflader (interfaces) og brugerflader. Når vi taler om UNIX og hermed POSIX miljøer er der følgende generelle holdninger som gør sig gældende, at bibeholde allerede eksisterende snitflader for at bevare flytbarheden og at tilføje nye snitflader for at udstyre systemet med nye egenskaber. Hvor et interface definerer hvad noget gør, udtrykker en implementering hvorledes noget gøres. Implementeringer er funktionelle, usynlige og kan derfor ændres uden at påvirke programmer eller brugerne i negativ retning.



Forord til POSIX standarden

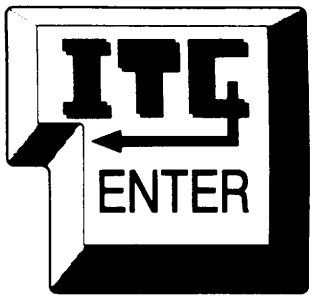
Formålet med POSIX standarden er at definere standardiseret interface for operativ systemet og dettes miljø. Idag tager POSIX standarden sit udgangspunkt i UNIX operativ systemet og fokuserer primært på "C" programmeringssprogsinterface til operativ systemet. Denne standard er hovedsagelig beregnet til brug for system implementatorer og applikationssoftware udviklere.

I skrivende stund er 13. udgave af dokumentet (IEEE std. 1003.1–1988) "Portable Operating System Interface for Computer Environments" sendt til ISO's Centrale Sekretariat for registrering som Draft International Standard (DIS) med dertil hørende 6 måneders afstemningsperiode blandt medlemmer af ISO. Den kommende standard vil bære ISO nummer IS 9945:1988 og er kun første del af flere relaterede standarder.

Den første del af standarden beskæftiger sig med kerne relaterede interfaces. Af andre projekter som varetages af samme gruppe af eksperter kan nævnes følgende:

- Shell and Utility faciliteter
- Verifikationstester
- Real-tids faciliteter
- Ada programmeringssprog bindinger
- Sikkerhed
- Programmeringssprog uafhængige service beskrivelser
- FORTRAN programmeringssprog bindinger
- Netværk interface faciliteter
- System administrationen

Arbejdet forbundet med standardiseringen af POSIX varetages i ISO's regi af Join Technical Committee 1 (JTC 1), Subkomite 22 (SC 22),



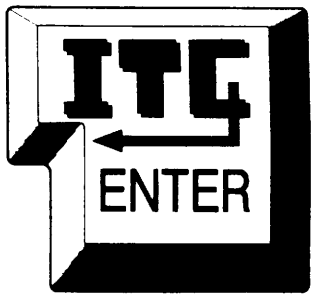
Arbejdsgruppe 15 (WG 15) – POSIX. I Danmark er der etableret arbejdsgruppe i Dansk Standardiseringsråd's regi under Fagudvalg 8 (FaU 8), Hovedudvalg 142 (S 142 – Informations Teknologi), Underudvalg 22 (u 22 – Programmeringssprog og Operativ Systemer), Arbejdsgrupper 11, 12 og 13 (A 11, A 12 og A 13 – POSIX og programmeringssprog "C"). Disse arbejdsgrupper tæller idag ca. 15 eksperter fra alle dele af dansk erhvervsliv både det offentlige og det private samt forskningsinstitutioner.

Selve POSIX dokumentet er idag opdelt i fire hoveddele:

1) Beskrivelse af omfanget af standarden (Kapitel 1)
2) Definitioner og globale koncepter (Kapitel 2)
3) De forskellige interface faciliteter (Kapitler 3–9)
4) Data udvekslingsformater (Kapitel 10)

Dokumentet er hovedsagelig baseret på følgende materialer:

UNIX Seventh Edition (V7)
UNIX System III
UNIX System V
4.2 Berkeley Software Distribution (4.2BSD) og
4.3 Berkeley Software Distribution (4.3BSD)

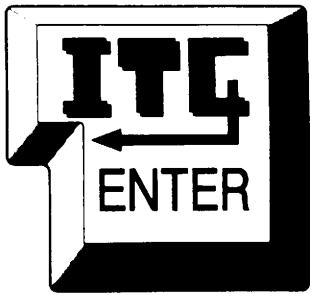


Standardens formål

POSIX standarden definerer standardiseret interface og miljø for support af portabiliteten (flytbarheden) af applikationer på kilde tekst niveau.

Til at begynde med fokuserer man i standarden på de servicer som "C" programmeringssprog interface kan supportere. Fremtidige versioner (revisioner) vil indeholde bindinger til andre programmeringssprog såvel som til "C". Dette vil indebære at standarden på et vis tidspunkt vil blive splittet i to dele. Den ene som vil beskrive krav til den indre del ('core') og som vil være uafhængig af programmeringssprogene, samt den anden del som vil indeholde de relevante sprogbindinger. 'Core' delen vil definere set af service krav som vil være fælles for hvilket som helst programmeringssprog og som kan bindes til denne standard. Disse servicer vil blive beskrevet i termer som funktionelle krav og vil ikke definere programmeringen af sprog afhængig interface.

Denne standard beskriver eksterne karakteristika og faciliteter som er vigtige for applikationsudviklere. Standarden beskriver ikke de interne konstruktionsteknikker som er nødvendige for at opnå disse karakteristika. Specielt fokuserer standarden på disse funktioner og faciliteter som er nødvendige for at støtte bred variation af kommercielle applikationer. POSIX standarden er ekslusivt defineret på kilde tekst niveau. Formålet er at POSIX konforme applikationer på kilde tekst niveau kan blive oversat og eksekveret i POSIX konforme miljøer.



POSIX gruppe af standarder og de relaterede områder

POSIX standarden er tænkt som en del af standarder som vil supportere Open System Environment (OSE) og derfor skal opfattes som et komplement til andre allerede eksisterende standarder eller standarder som er under udvikling. Disse standarder kan deles i to grupper, den ene som er udviklet og er tilgængelig, og som supporterer OSE, samt den anden gruppe som er under udvikling og som vil få indflydelse på OSE.

Til at begynde med kan man nævne det fortsatte arbejde som POSIX gruppen vil lægge i definitionen af system interface. De fremtidige mål for gruppen vil være at definere:

- programmeringssprog uafhængige specifikationer
- udvidet og homogen data udveksling (interchange) format
- implementeringen af flere forskellige funktioner efterhånden som standarden udvikler sig

Den næste standard med meget tæt tilknytning til POSIX er kommende standard for programmeringssproget "C". På nuværende tidspunkt ligger "C" meget tæt til at blive registreret som blivende standard. Disse to arbejdsgrupper har meget tæt samarbejde med hinanden, eftersom POSIX standarden som den er foreslået idag, gør meget brug af "C".

Som beskrevet tidligere pågår der mange parallelle aktiviteter i forbindelse med standardiseringen af POSIX, bl. a. den næste stor standard med hensyn til POSIX vil omfatte Shell and Utilities (IEEE 1003.2). Den foreslåede standard definerer på kilde tekst niveau, interface til 'skal'-servicer (shell) og fælles platform (utility) af programmer som er konforme

med POSIX kerne interface (IEEE 1003.1). Formålet med denne standard er defineret på følgende måde:

> At specificere standard interface som kan udnyttes fælles af både applikationsprogrammer og brugere af terminal-kontrollerede programmer for at give bedre service af mere kompleks struktur end den primitive som er supporteret af kerne standarden.

Denne standard skal inkludere følgende komponenter:

1) Applikationsprogram primitiver for specifikation af implementeringsdefinerede instruktioner for shell faciliteter.

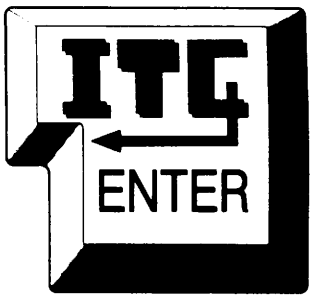
2) Standardiseret kommando sprog for shell som inkluderer program eksekution, Input/Output omdirigering og 'pipeling' (datakanal håndtering), håndtering af argumenter, variable substitutioner og udvidelser, samt serie af kontrol konstruktioner som er kendte fra andre højniveau strukturerede programmeringssprog.

3) Anbefale kommando syntaksen for kommando navne og argument specifikation.

4) Primitiver som applikationsprogrammer og shell-sproget kan benytte til at fortolke og analysere (pärsing) kommando argumenter.

5) Anbefale miljø (environment) variable for brug i shell-skripter (programmer) og applikationsprogrammer.

6) Definition af platform (utilities) som kan kaldes fra applikationsprogrammer for komplekse data manipulationer og andre fælles opgaver for applikationer.



7) Platformer og standarder for installation af applikationer.

Til gengæld kommer den kommende standard ikke til at omfatte følgende områder:

1) Operativ systemets administrative kommandoer.

2) Kommandoer som er nødvendige for installationen, configureringen og vedligeholdelsen af operativ systemet og fil-systemet.

3) Netværk kommandoer.

4) Terminal kontrol eller bruger interface programmer.

5) Grafiske kommandoer eller interfaces.

6) Tekst formatteringsprogrammer eller sprog.

7) Database programmer eller interfaces.

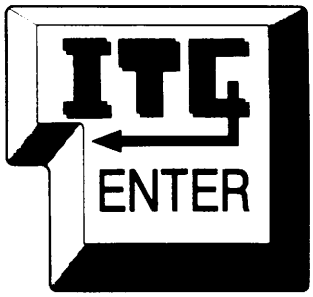
Arbejdsgruppen er i øjeblikket igang med at udarbejde forslag til konformitetstester og verificering af standarder som vedrører POSIX (IEEE 1003.3).

Et andet område som POSIX standardiseringsgruppe er igang med at lave standard er real-tids udvidelser (IEEE 1003.4). Denne standard (som i fremtiden eventuelt skal indgå i kerne standarden), vil udvikle ekstentioner til kerne standarden med henblik på opbygning af de nødvendige interfaces som behøves for portabelt realtid applikation. Målet for standarden vil være at inkludere følgende områder som relateres til realtidsproblematikker:

1) Prioritetsskedulering.

2) Semafor styring.

3) Contiguous (tilstødende) filer.



4) Mellem-process **message passing.**

5) Resultat (event) notifikation.

6) Memory lockning.

7) Asynkron I/O.

8) Synkron I/O.

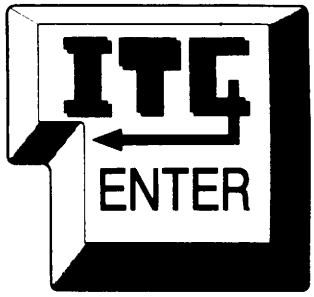
9) **High resolution timers.**

Derudover eksisterer følgende projekter indenfor IEEE (The Institute of Electrical and Electronics Engineers), 1003.5 Ada programmeringssprog bindinger, 1003.6 Sikkerhedsaspekter, 1003.7 System administration samt 1003.0 Open System Guidelines. Læg mærke til at på nuværende tidspunkt kun projekt 1003.1 Kerne interfaces er nået til DIS stadiet, de andre projekter har forholdsvis lang tid endnu inden de kan blive registreret som international standard, den der ligger tættest på er 1003.2 Shell og Utilities.

Af de andre og relaterede standarder kan nævnes følgende som vil kunne have tæt forbindelse med den standardisering som finder sted i POSIX gruppen:

I) Netværk standarder:

* OSI modellen – IS 7498
* Lag 1 – IS 8802/X
* Lag 2 – IS 8802/2
* Lag 3 – IS 8348, 8473, 7777
* Lag 4 – IS 8072, 8073
* Lag 5 – IS 8326, 8327



* Lag 6 – IS 8822 (DP), 8823 (DP)
* Lag 7 CASE – IS 8649 (DP), 8650 (DP)

  FTAM – IS 8571 (DP)

  MHS (Mail) – CCITT X.400

  Job Transfer – IS 8831 (DP), 8832 (DP)
* Wide Area Net – CCITT X.25

II) Programmeringssprog standarder:

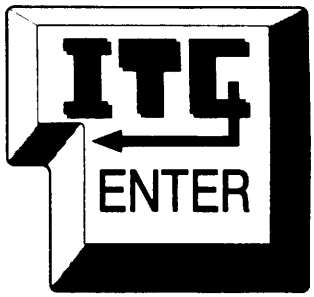
* Ada
* Basic
* COBOL
* FORTRAN
* Pascal

III) Grafiske standarder:

* GKS (Graphical Kernel System) både 2-D og 3-D
* PHIGS (Programmers Hierarchical Interactive Graphics System)
* CGM (Computer Graphics Metafile)

IV) Databasesprog standarder:

* NDL (Network Database Language)
* SQL (Structured Query Language – relational database)

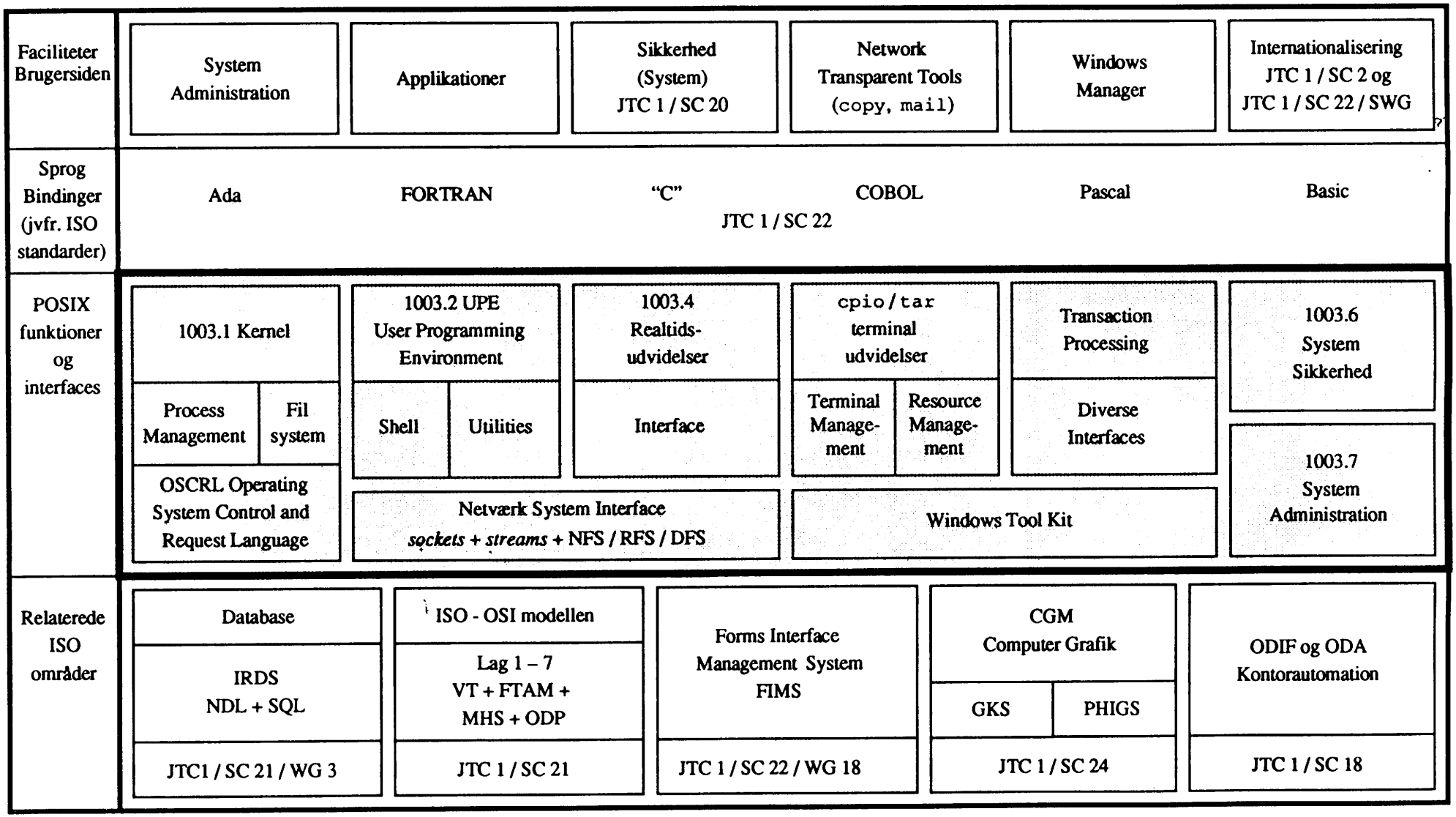


Af andre standardiseringsprojekter som er blevet etableret fornylig og som kunne tænkes at få indflydelse på standardiseringen af POSIX kan nævnes følgende:

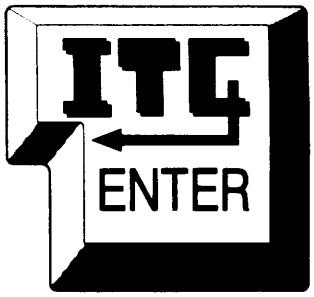
* ODP (Open Distributed Processing – del af OSI)
* EDI (Electronic Data Interchange)
* VT (Virtual Terminal – del af OSI)
* FIMS (Forms Interface Management System)

Derfor er de eksperter som er igang med at definere den første standard med hensyn til operativ systemet nødt til i det mindste at holde sig orienteret om de standarder som ligger tæt på den standard som de er igang med (POSIX).

POSIX standardens omfang og dens interfaces til andre områder



Systemets Sikkerhed, internationalisering og distribuerede processer er globale problemer, som skal løses i forbindelse med både POSIX funktioner og faciliteter på bruger siden.



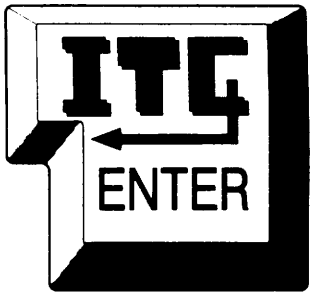
POSIX beskrivelse af standarden

POSIX arbejdet er blevet startet af IEEE i USA og efterhånden er dette arbejde spredt til at involvere ISO, dette skete da USA's repræsentant i ISO – American National Standards Institute (ANSI) foreslog POSIX standardiseringen spredt over alle medlemsnationer af ISO. På verdensplan varetages dette arbejde af flere hundreder eksperter med tæt tilknytning til UNIX-verdenen. Disse eksperter repræsenterer alle afsnit af samfundet som måtte have interesse i denne standard heriblandt kan der nævnes repræsentanter for hardware leverandører, leverandører af operativ systemer og andre software applikationer, software designere, systemintegratorer og konsulenter, forskere, foreninger og mange andre.

Konceptuelt beskriver standarden et sæt af fundamentale servicer som eksperterne mener er nødvendige for effektiv konstruktion af applikationsprogrammer. Adgangen til disse servicer er muliggjort ved hjælp af definitioner af interface, som idag bruger "C" programmeringssprogets syntaks og semantikker. Siden denne interface tillader at man udvikler portable applikationer blev standarden døbt POSIX (Portable Operating System Interface), som i sin tid foreslået af Richard Stallman.

Den potentielle gruppe af brugere til denne standard kan findes blandt alle personer som er involveret i bredt industri standard baseret på UNIX systemet, blandt disse grupper kan man fremhæve specielt følgende:

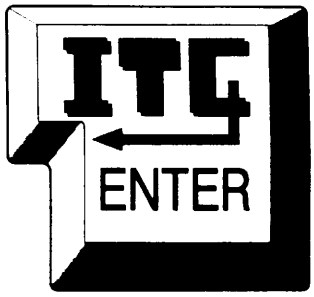
- personer involveret i indkøb af både hardware og software systemer
- personer med ansvaret for at definere fremtidige strategier mht. virksomhedernes brug af Informations Teknologi
- operativ system implementatorer og



- systemfolk som udvikler applikationer hvor portabiliteten er formålet

Arbejdsgruppen omkring POSIX har koncentreret sine bestræbelser på additioner og afklaringen samt ændringer i forhold til UNIX systemet fra hvilket POSIX er ekstraktet og derfor omfatter standarden ikke UNIX systemet i sin helhed, eftersom gruppen havde mandat til at anvende eksisterende operativ system og ikke udvikle et nyt.

I standarden er materiale som er "udenfor" standarden eller ikke defineret i standarden at betragte som implementerings-defineret hvis dette materiale bør indeholdes i implementeringen.

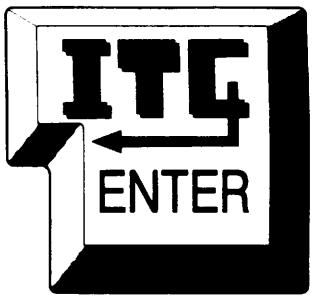


De offentliges satsning på standardiserede miljøer

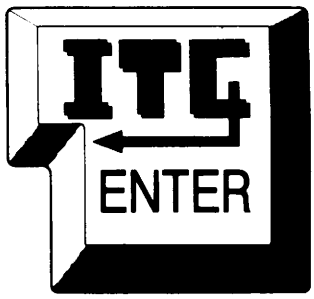
Over hele verden kan man idag mærke en stor interesse fra det offentlige marked til at satse på standardiserede produkter, det ville sige produkter som opfylder de internationale standarder. Specielt lægger de offentliges indkøbsinstanser vægt på at udstyret som skal bruges i forbindelse med informations teknologi opfylder standarder som er relaterede til det såkaldte åbne miljø. Dette skyldes at de offentlige i fremtiden vil udnytte de samme applikationer på udstyret som bliver leveret fra mange forskellige hardware leverandører.

Denne trend har affødt at mange applikationsudviklere idag satser på at udvikle programmer som forholdsvis nemt kan flyttes imellem forskelligt maskinel. En af de vigtigste standarder på dette felt er POSIX standarden som i sin helhed skal yde support til portabiliteten af applikationer.

Som nogle eksempler på denne udvikling kan nævnes at det amerikanske NIST som bestemmer hvilke standarder skal overholdes af de leverandører som leverer produkter til det amerikanske statsadministration, har bestemt at POSIX er en af de standarder som administrationen har defineret, at leverandørerne skal overholde. De amerikanske standarder som gør sig gældende indenfor statsadministrationen bliver løbende opdateret og indgår i en liste af såkaldte FIPS-standarder. POSIX er en af disse standarder, af andre lignende standarder som er listet i FIPS kataloget kan nævnes: ISO-OSI modellen for netværk, SQL sproget for relationelle databaser og lignende. Derudover har det amerikanske NIST institut opbygget en konformitet test med hensyn til aftestningen af POSIX standarden, denne test bygger dog på "Draft 12" af den foreslåede standard, men NIST har sagt at ligeså snart standarden for POSIX bliver vedtaget som international standard – IS, vil de justere deres tester til at dække den nye standard.



Med hensyn til Europa så kan vi spore samme trend som i USA. Kommissionen som koordinerer standardiseringen i EF med hensyn til informationsteknologi har fornylig placeret ordre hos CEN/CENELEC på en europæisk version af POSIX standarden. EF søger at gennemføre en synkronisering af standarder for medlemslandene, derudover har Kommissionen etableret såkaldt "Conformance Test Service" (CTS) ordning for de leverandører som vil levere standardiserede produkter til den offentlige forvaltning i EF's medlemslande. EFTA landene har frivilligt tilsluttet sig denne ordning, det ville sige at disse lande frivilligt har forpligtet sig til at støtte samme standarder som EF landene vedtager for deres vedkommende. Igen en af de standarder som EF satser på er POSIX standarden og som det fremgår af Tillæg 4, hvor der gengives ordre fra Kommissionen til CEN/CENELEC, skal den europæiske version af standarden (EN) være færdig i april 1990.



POSIX målsætninger og perspektiver – en opsummering

* POSIX, som defineret i øjeblikket, er et afgørende første step i retning af support for portabiliteten af applikationsprogrammer

* Den nuværende definition af standarden, må blive udvidet med større funktionalitet som:

  – Shell og tools

  – System administrationen

  – Terminal interface udvidelser

  for at skaffe fuld operativ system funktionalitet

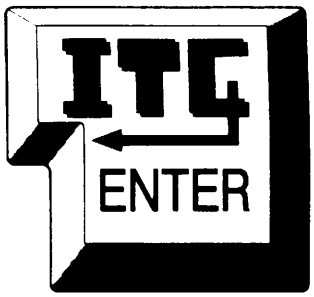
* Selv udvidet POSIX vil ikke være tilstrækkelig for at opnå fuld portabilitet af alle applikationer

* Man kan idag finde større forståelse for nødvendigheden af at finde en arkitektonisk løsning med hensyn til applikationernes portabilitet

* Denne systemarkitektur som er krævet for at opnå portabilitet må indeholde den bedste funktionalitet som kan supportere det bredeste udsnit af applikationer

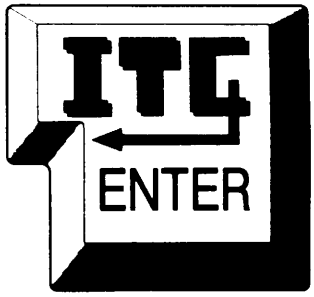
* De funktionelle komponenter af en sådan arkitektur bør udgøre et sæt af standardiserede "tool box'e" som kan bruges i udviklingen og vedligeholdelsen af portable applikationer

* Denne arkitektur skal baseres på "Open Systems" koncepter og skal udvikles som en integreret samling af åbne standarder



* Applikationsportabilitetsproblemer som skal løses:
  - bestemmelse af de funktionelle karakteristika for portabel arkitektur
  - ikke leverandør afhængige specifikationer
  - udvikling af behørige bindinger
  - leverandørernes forpligtelse til at bruge specifikationer i udviklingen af produkter
  - brugernes forpligtelse til at bruge specifikationer i indkøbsaftaler
  - konformitetstester

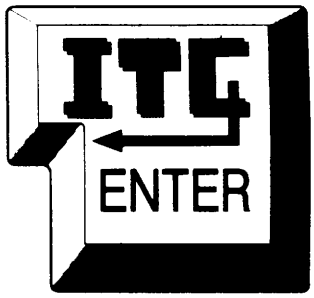
POSIX standarden går, sammen med en række af andre standarder som er igang med at definere det åbne miljø, en stor fremtid i møde.



Dansk POSIX guide

Del 2

Lidt teknik omkring standarden



Teknisk gennemgang af POSIX standarden

Arbejdet med POSIX standarden har medført mange overvejelser hos de eksperter som deltager aktivt i definitionen af standarden. Her kan nævnes nogle af de mål som gruppen har defineret som udgangspunkt for sit arbejde.

Applikationsorienteret

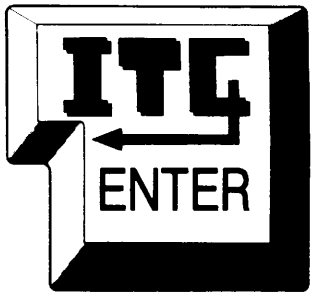
Fundamentet for gruppens arbejde er at fremme portabiliteten af applikationsprogrammer over et stort udsnit af UNIX baserede systemer ved hjælp af udviklingen af klar, konsistent og utvetydig standard for interface specifikationen for portabel operativ system baseret på UNIX systemets dokumentation. Gruppen prøver ikke på at udvikle et nyt system interface men baserer sit arbejde på fælles, eksisterende definitioner af UNIX systemet.

Interface og ikke implementering

Standarden definerer interface og ikke hvordan implementationen skal se ud. Man har ikke skelnet til forskelle mellem biblioteksfunktioner og systemkald, begge benævnes som funktioner. Der gives ingen detaljer om hvordan en funktion skal implementeres. Symbolske navne er givet for konstanter (som **signaler** eller **error koder**), fremfor numeriske værdier.

Kildetekst og ikke objektkode portabilitet

Standarden er blevet udviklet med det formål, at et program som er skrevet og oversat for eksekvering i en konform implementering også kan oversættes for eksekvering i en anden konform implementering.
Standarden giver ingen garanti for at eksekverbar (objekt eller binær) kode vil kunne "køre" i en anden konform implementering end den, den er blevet oversat på.



Ingen Superbruger, ingen systemadministration

Den første del af standarden, som omtalt tidligere omfatter beskrivelse af operativ systemets interface og derfor omfatter denne del ikke aspekter af system administrtionen. Som nævnt tidligere har IEEE startet et nyt projekt som skal udforme standarden mht. dette emne. I øjeblikket diskuteres der blandt ISO's medlemmer om denne del også skal inkluderes i kommende arbejde i ISO's regi.

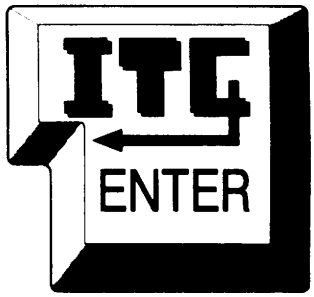
Minimal interface, minimal definition

POSIX standarden skal opfattes som en minimal standard (af hensyn til konformitetstester) og derfor, eksempelvis specificeres i standarden kun et set af funktioner for at implementere en given funktionalitet. Undtagelser herfra kan findes i nogle tilfælde hvor lang tradition og mange eksisterende applikationer inkluderer en bestemt funktionalitet, som for eksempel *creat()*. I sådanne tilfælde, og igennem hele standarden, er redundante definitioner undgået. *creat()* er defineret som speciel tilfælde af *open()*. Redundante funktioner eller implementationer med mindre tradition er blevet ekskluderet fra standarden, for eksempel er *seekdir()* og *telldir()* ikke inkluderet i "**Directory Operations**" afsnittet.

Muligheder for bred implementering

Arbejdsgruppen har bestræbt sig på, at gøre det muligt at alle funktioner kan implementeres på de fleste eksisterende og potentielle systemer. Dette inkluderer:

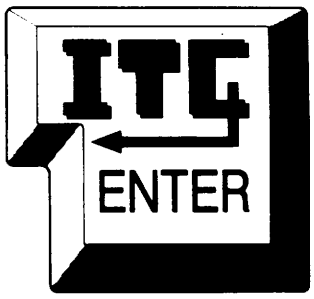
- alle eksisterende systemer som er videreudviklet på baggrund af AT&T's kode (Version 7 eller senere)
- kompatible systemer som ikke nødvendigvis er baseret på AT&T koden



- emulationer af UNIX systemer som "kører" på helt andre operativ systemer
- netværkssystemer
- distribuerede systemer
- systemer som bruges på en stor vifte af hardware

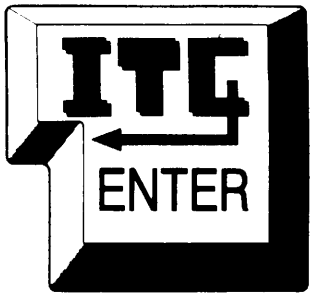
Minimale ændringer i forhold til historiske implementationer

Der eksisterer ikke kendte implementeringer af UNIX systemet som ikke vil behøve nogle ændringer for at opnå konformiteten med standarden. I nogle tilfælde passer standarden ikke med de eksisterende system interfacer. Alligevel er der i standarden, komplet sæt af funktioner, typer, definitioner og koncepter som former en interface som er fælles for de fleste kendte implementeringer. Antallet af de nødvendige ændringer er holdt på et minimum men de eksisterer. Standarden har ingen præferencer i forhold til nogle bestemte leverandørers produkter. Den ligner UNIX systemet, men er ikke identisk. Ordet UNIX er ikke anvendt i standarden af denne grund og fordi dette ord er beskyttet handelsmærke af en kendt leverandør. Man skal lægge mærke til at kommercielle implementeringer af POSIX standarden vil have forskellig grad af udvidelser. Nogle af disse udvidelser beror på "historisk anvendelse" og vil blive brugt for eksekveringen af eksisterende applikationer, disse funktioner bør efterhånden aftage og vil blive afløst af de standardiserede funktioner, efterhånden som applikationer fornyes. Nogle udvidelser vil bevæge sig udenfor det område som dækkes af POSIX standarden, disse funktioner bør bruges med stor omhu sådan at det vil være mulig at adoptere fremtidige udvidelser af POSIX standarden, og/eller porte til implementationer som supporterer disse servicer på en anden måde.



Minimale ændringer til eksisterende applikationskode

Arbejdsgruppen omkring POSIX standarden ønsker mindst muligt arbejde for udviklere af applikationsprogrammer og ikke mere. Alligevel skal alle kendte implementeringer af operativ systemets interface ændres af hensyn til konformiteten (som beskrevet tidligere), og derfor vil det være nødvendigt for nogle af applikationsprogrammerne at foretage mindre ændringer.



Basis POSIX dokumentation

POSIX arbejdsgruppen har i sit arbejde med standarden taget som udgangspunkt 1984 udgaven af "/usr/group Standard" dokumentet, bedre kendt som det første arbejdsdokument for POSIX. Dette dokument tager udgangspunkt i programmeringsinterface for System III og er stamfader for Library sektionen af "C" standarden.

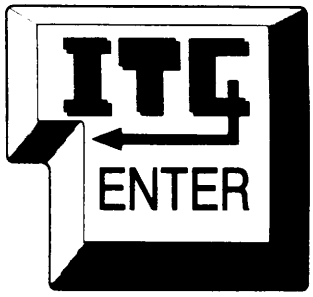
Eftersom POSIX standarden idag er meget tæt knyttet til "C" programmeringssprog, har gruppen benyttet sig af "C" standardiseringsgruppens arbejdsdokumenter ("programming Language C Standard", som forventes snart at blive registreret som DIS).

Ligeledes blev "X/OPEN Portability Guide" benyttet, fordi dette dokument tager højde for mange af de portabilitetsaspekter som POSIX gruppen kunne drage nytte af, selvom definitioner oftest er lidt forskellige.

Af historiske grunde bør man nævne følgende dokumenter som man har anvendt i sit arbejde:

- "UNIX Time-Sharing System: UNIX Programmer's Manual, Seventh Edition" (Version 7)
- "UNIX System III Programmer's Manual"
- "AT&T System V Interface Definition (SVID)", udgave 2, bind 1 til 3
- "4.3 Berkeley Software Distribution, Virtual VAX-11 Version (4.3BSD) Manuals"

UNIX systemet har ændret sig meget siden **/usr/group** har publiceret sit første "Standard" dokument. Det er kommet flere varianter og former til siden dengang. Derfor blev nuværende standard reorganiseret og



omformatteret siden sin første udgave (**Draft 1**) til det registrerede dokument (**Draft 13**). Lejlighedsvis nævner man Ottende og Niende (Eight og Ninth) Editioner som er efterfølgere af Version 7 fra Bell Laboratorier forskningssystem, sammenhængen er relateret til `streams` eller mellem-process kommunikationen som ikke er indeholdt i standarden, men har indflydelse på diskussioner omkring mekanismer ved inter-process kommunikationen.

Mens man arbejdede med POSIX dokumentet var version 4.2 af BSD den mest benyttede, men standarden refererer til 4.3BSD i stedet for, fordi de afgørende forskelle ligger i bedre ydelse (performance) og 4.3BSD manualer beskriver 4.2BSD bedre end 4.2BSD manualer gør. Man henviser, i dokumentet, aldrig til System V manualer, fordi disse er mere definitive. Men alligevel er standarden tættere på SVID end til et hvilketsomhelst andet dokument.

Specifikke afledninger

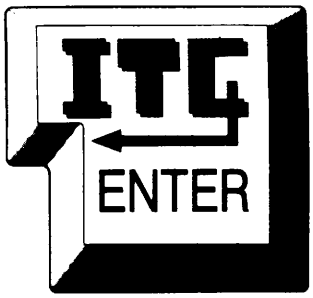
Nogle områder af standarden er klart afledt fra faciliteter som er indeholdt i specifikke systemer. Hermed følger en liste over de mest vitale områder af standarden samt hvilket system de stammer fra.

FIFO – FIFO speciel fil faciliteten eksisterer i System III, /usr/group Standard og System V, men ikke i Version 7, 4.2BSD eller 4.3BSD.

reliable signals – kan relateres til 4.3BSD modellen, disse blev introduceret i forbindelse med problemer omkring pålideligheden af signaler.

job control – er afledt fra 4.3BSD.

saved set-user-ID (eller **saved set-group-ID**) – denne optionale funktion, hovedsageligt brugt i `exec` og **Set User** og **Group ID** er afledt fra System V.



supplementary groups – er defineret som en enkelt gruppe per proces, som i System V, men **User Identification** (specielt *getgroups()*) tillader, som en option, flere grupper per process i lighed med 4.3BSD.

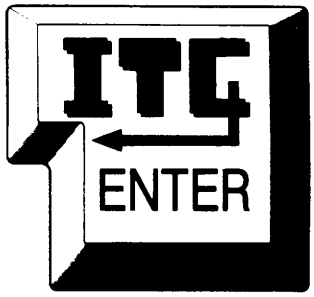
uname() – funktionen stammer fra /usr/group Standard, som har taget denne fra System III og eksisterer stadig i System V, denne funktion eksisterer ikke i Version 7 eller 4.3BSD.

opendir(), readdir(), rewinddir(), closedir() – funktioner som beskriver **Directory Operations** er afledt fra 4.2BSD og blev senere inkluderet i System V Release 3.

mkdir(), rmdir(), rename() – disse funktioner kommer fra 4.2BSD, med undtagelse af *rename()* er disse inkluderet idag i System V Release 3.

`termios` – strukturen som er beskrevet i standardens **Device-** og **Class-Specific Functions**, ligner mere funktioner som er indeholdt i System V end 4.3BSD, men alligevel er denne struktur ikke helt kompatibel med nogen eksisterende implementering. POSIX gruppen kunne ikke idag finde nogle implementeringer som i tilstrækkelig grad kunne yde support til problemer omkring håndteringen af international karakter sæt, hurtige interfacer og netværk.

archive format – arkiveringsformatet afledte mange diskussioner i arbejdsgruppen om hvilket format man skulle vælge til brug i standarden og derfor eksisterer der to formatter som begge er en del af standarden. Udvidet `tar` formatet er taget fra programmer brugt af Version 7 og 4.3BSD, disse supporteres også af System V, mens udvidet `cpio` format er taget fra System V specifikationen.



Definitioner og generelle krav

Ved læsningen af standard dokumentet bør man være opmærksom på følgende definitioner.

Terminologi

implementering-defineret (implementation-defined) – en værdi eller en adfærd er implementering-defineret hvis implementeringen definerer og dokumenterer de krav som er nødvendige for korrekt konstruktion af programmer og korrekte data.

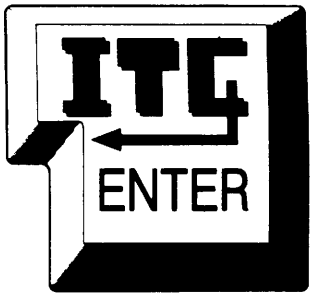
må (may) – med hensyn til implementeringen kan ordet "må" tolkes som at pågældende funktion ikke er nødvendig for at opfylde standardens krav men kan supporteres af implementeringen, af hensyn til konformitetstester funktioner som bruger ordet "må" skal ikke bruges til at validere konformiteten.

skal (shall) – i POSIX standarden brugen af ordet "skal" bør tolkes som absolut krav til implementeringen og vil indgå i konformitetstester.

bør (should) – ved implementeringen af standården skal ordet "bør" opfattes som en implemmenteringsanbefaling og ikke som krav.

supporteret (supported) – nogle af funktionaliteter i POSIX standarden er optionale, men interfacen til disse funktionaliteter er altid påkrævet. Hvis funktionaliteten er "supporteret" skal interfacen fungere som beskrevet i standarden, hvis funktionaliteten er ikke "supporteret" skal interfacen altid returnere en indikation som er specificeret for denne situation.

ikke-defineret (undefined) – en værdi eller en adfærd er "ikke-defineret" hvis standarden ikke kræver portabiliteten for applikationen, implementeringen eller andre standarder må specificere resultater af brugen af denne værdi eller adfærdet.



ikke-specificeret (unspecified) – definieres på samme måde som "ikke-defineret".

Konformitet i forhold til standarden

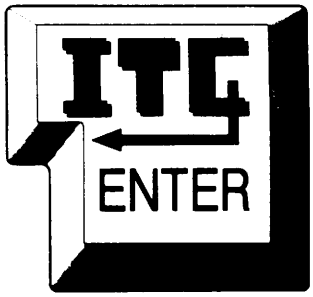
Konform implementering skal møde alle nedennævnte kriterier:

- Systemet skal yde support for alle krævede interfacer som er definerede i denne standard, disse interfacer skal supportere den funktionalitet som er beskrevet i standarden.

- Systemet kan supportere ekstra funktioner eller faciliteter som ikke er påkrævet af denne standard. Ikke standard udvidelser skal defineres som sådanne i system dokumentationen. Ikke standard udvidelser, hvis brugt, kan ændre adfærden af funktioner eller faciliteter som er defineret i standarden, i sådanne tilfælde skal system dokumentationen definere miljøet i hvilket disse applikationer kan "køre" med den adfærd som er defineret i standarden.

Dokumentation som kræves for konforme implementeringer

Med hver konform implementering af POSIX standarden skal leverandøren af implementeringen, levere dokumentationen som skal have samme struktur som standarden. Dokumentationen skal beskrive indeholdet af **<limit.h>** og **<unistd.h>** header filer, samt beskrive værdier og under hvilke konditioner disse værdier kan skifte, samt eventuelle begrænsninger af sådanne variationer. Dokumentationen skal derudover beskrive adfærd implementeringen med hensyn til alle "implementering-defineret" faciliteter som er identificeret i standarden, dokumentationen behøver ikke at beskrive faciliteter som er "ikke-defineret" eller "ikke-specificeret".



POSIX – generelle termer

Følgende afsnit definerer de termer som er særegne for standarden.

access mode – formen som bestemmer adgangstilladelser til en fil

adress space – hukommelse (memory) lokationer som processer kan henvise til

appropriate privileges – implementering-defineret, process tilknyttede privilegier med hensyn til funktionskald og funktionskaldsoptioner som defineret i standarden, og som kræver særlige privilegier

background process group – alle process grupper som hidrører fra en session som er blevet startet i forbindelse med en kontrol-terminal som ikke er i forgrund process gruppen

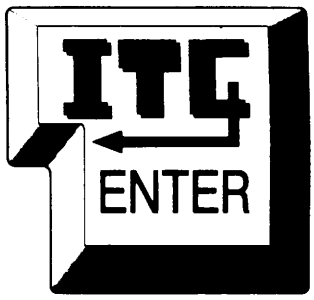
block special file – en fil der henviser til en device, blok speciel-fil er normalt forskellig fra karakter speciel-fil ved at støtte adgangen til device'n på en sådan måde, at hardware karakteristika af device'n ikke er synlige.

C Standard – en forkortelse af "Programming Language C Standard"

character – en sekvens af en eller flere bytes som repræsenterer enkelt grafisk symbol (glyph/tegn)

character special file – en fil der henviser til en device, en specifik karakter speciel-fil er **terminal device fil** hvis adgangskriterier er defineret i **"General Terminal Interface"** afsnittet, andre karakter speciel-filer har ingen struktur defineret i standarden og er derfor implementering-defineret

child process – se **process**



clock tick – nummer af intervaler i sekundet, defineret af [CLK_TCK], brugt til at udtrykke værdien i typen *clock_t*

controlling process – sessionsleder som etablerer forbindelsen til kontrollerende terminal, skulle terminalen ophøre med at være kontrollerende terminal for denne session, skal sesionsleder ophøre med at være kontrollerende process

controlling terminal – terminal som er tilknyttet en session, alle sessioner må i det mindste have en kontrollerende terminal i forbindelse med eksakt en session, bestemte input sekvenser fra kontrollerende terminal bevirker at `signal`'er sendes til alle processer i samme process gruppen som er i tilknytning til denne kontollerende terminal

current working directory – se **working directory**

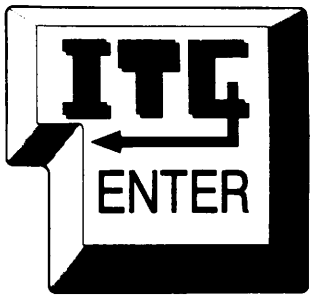
device – computer ydreenhed (peripheral) eller et objekt som fremstår overfor en applikation som sådan

directory – en fil som indeholder **directory entries**, ingen af to **directory entries** i samme **directory** skal have samme navn, **directory** kan oversættes til katalog

directory entry (eller **link**) – et objekt som er forbundet med et fil-navn og gennem dette med en fil, forskellige **directory entries** kan forbinde fil-navne med samme fil

dot (. punktum) – fil-navn bestående af en enkel punktum karakter (.), brugt i **pathname resolution**

dot-dot (.. to punktummer) – fil-navn bestående udelukkende af to punktum karakterer (..), brugt i **pathname resolution**



effective group ID – et kendetegn af en process som bliver brugt til at fastsætte forskellige tilladelser herunder adgangen til filer, denne attribut/værdi ændres i løbet af processens levetid, som beskrevet i *setgid*() og `exec`, se yderligere **group ID**

effective user ID – et kendetegn af en proces, ligner **effective group ID** og er beskrevet i *setuid*()

empty directory – et katalog som indeholder højst to **directory entries** nemlig **dot** (.) og **dot-dot** (..)

empty string (eller **null string**) – karakter række (array) hvis første element er en **null** karakter

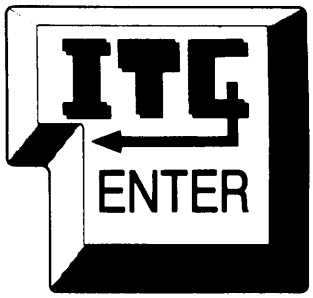
Epoch – tids-konstant svarende til 0 timer 0 minutter 0 sekunder, den 1. januar 1970 CUT (Coordinated Universal Time), se yderligere **seconds since the Epoch**

feature test macro – et `#define`'d symbol brugt til at bestemme inkluderingen af definerede faciliteter fra **header** filen

FIFO special file (eller **FIFO**) – en fil, data skrevet til sådan en fil bliver læst på først-in-først-ud basis, andre karakteristika af FIFO er beskrevet i standarden under *open*(), *read*(),*write*() og *lseek*()

file – et objekt som det er tilladt at skrive til, læse fra eller begge dele, filen har sikkerhedsattributer som inkluderer adgangskontrol/tilladelser og type, standarden definerer flere forskellige fil typer som regulær fil, karakter speciel-fil, blok speciel-fil, FIFO speciel-fil og katalog (directory), andre fil typer kan defineres af implementationen

file description – se **open file description**



file descriptor – et pre-proces unikt, ikke negativt heltal (integer), som bliver brugt til at identificere en åben fil i forbindelse med fil-adgangen

file mode – et objekt indeholdende **file permission bits** (adgangskontrol) og andre fil karakteristika som beskrevet i standarden under **<sys/stat.h>**

filename – fil-navn bestående af 1 til [NAME_MAX] bytes brugt til at navngive en fil, navnet kan konstrueres af alle alfabetets tegn med undtagelse af "/" og **null** karakterer

file offset – en byte position i filen hvor den næste I/O operation starter

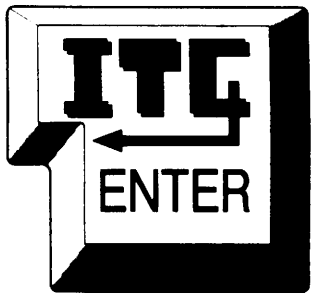
file permission bits – informationen om filen som bliver brugt, sammen med andre informationer, til at bestemme om en process har læse, skrive eller eksekvering/søge rettigheder til filen, disse bits er delt i tre grupper som vedrører ejeren, gruppen og alle andre og er indeholdt i **file mode** som beskrevet i standardens **<sys/stat.h>**

file serial number – identifikationsmærke som er unikt for en fil igennem hele fil systemet

file system – kollektion af filer og deres sikkerhedsattributer, samt plads for **file serial numbers** med referencer til disse filer

foreground process group – hver session som har etableret forbindelsen til kontrol terminal har nøjagtigt en process gruppe af sessionen som forgrund process gruppe af denne kontrollerende terminal, forgrund process gruppen har bestemmende privilegier ved adgangen til dennes kontrollerende terminal og som er forbudte for baggrund process gruppen

group ID – hver system bruger er medlem af mindst en gruppe, gruppen bliver identificeret ved hjælp af gruppens ID, som er et ikke negativt heltal (integer) og som kan blive inkluderet i objekter af typen *gid_t*



job control – en facilitet som tillader brugerne (selektivt) at stoppe eksekveringen af en process og fortsætte dennes eksekvering på et senere tidspunkt, brugerne anvender typisk denne facilitet ved hjælp af en interaktiv interface som supporterer terminal I/O driver og kommando fortolker (shell)

mode – kollektionen af alle attributer som specificerer fil typen og dennes adgangstilladelser

parent process – se **process**

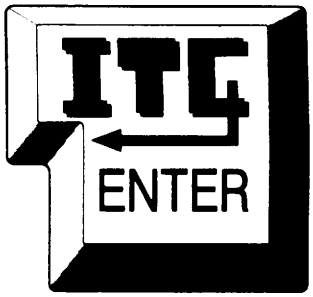
pathname – en streng som bruges for at identificere filen, denne streng består af maksimalt [PATH_MAX] bytes inklusive terminerende **null** karakter

pipe – et objekt med adgang for en ud af et par af **file descriptors** (fil-referencer) skabt af *pipe*() funktionen

process – en **address space** (adresseringsområde) og en simpel række af kontroller som eksekverer indenfor dette adresseringsområde ved brug af de nødvendige system resourcer, en proces startes af en anden proces ved hjælp af *fork*() funktionen, processen som starter *fork*() kaldes **parent** (fader) proces og processen som blive startet af *fork*() kaldes **child** (barn) proces

process group – hver proces i systemet er medlem af proces gruppen som bliver identificeret ved hjælp af **process group ID**, denne gruppering tillader signalering imellem relaterede processer

process group lifetime – en periode af tiden som starter med skabelsen af en proces gruppe og som slutter når den sidste proces i gruppen forlader denne, enten på baggrund af termineringen (afslutningen) eller som følge af *setsid*(), eller *setpgid*() funktioner



process lifetime – efter at processen er startet ved hjælp af *fork*() funktionen forbliver processen aktiv, dennes række af kontroller og adresseringsområde eksisterer indtil processen terminerer, når dette indtræder overgår processen til et inaktivt stadie hvor nogle resourcer som processen brugte bliver givet tilbage til systemet, nogle af processens resourcer som f.eks. **process ID** er stadig i brug, når en anden process eksekverer *wait*() eller *waitpid*() funktionen overfor en inaktiv proces bliver de resterende resourcer overdraget til systemet, den sidste resource som processen returnerer til systemet er **process ID** og på dette tidspunkt slutter processens levetid

seconds since the Epoch – er en værdi som skal fortolkes, som et antal af sekunder mellem specificeret tidspunkt og **the Epoch**, Coordinated Universal Time (CUT) specificeret i termer af sekunder (*tm_sec*), minutter (*tm_min*), timer (*tm_hour*), dage siden 1. januar af et år (*tm_yday*) og kalenderår minus 1900 (*tm_year*), dette antal af sekunder defineres som sekunder siden **the Epoch**, værdien kan udregnes fra følgende formel:

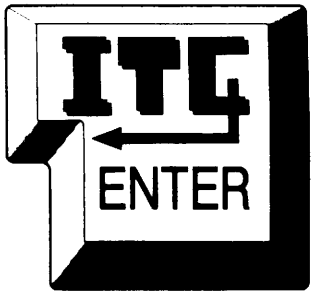
$$tm\_sec + tm\_min \times 60 + tm\_hour \times 3600 + tm\_yday \times 86400 + (tm\_year - 70) \times 31536000 + ((tm\_year - 69) / 4 \times 86400)$$

session – hver proces gruppe er medlem af en session, processen opfattes som et medlem af sessionen hvor dennes proces gruppe er medlem af, en proces når den bliver startet tilhører automatisk den samme session som processens skaber

signal – er en mekanisme som bruges til at underrette processer om begivenheder som hænder i systemet som f.eks. hardware problemer

slash – karakter "/", denne karakter er kendt som **solidus** i ISO standarden IS 8859/1

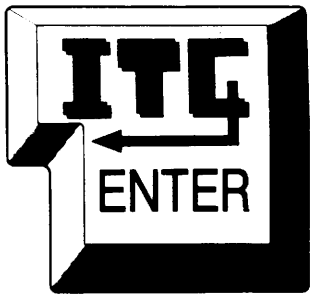
system – en implementation af POSIX standarden



system process – en anden proces end den der eksekverer applikation og som er defineret af systemet og har **process ID**

user ID – hver bruger af systemet bliver identificeret af ikke negativt heltal (integer) som kan blive indeholdt i et objekt af typen *uid_t*

working directory (eller **current working directory**) – et katalog (directory) tilknyttet en proces som er brugt i **pathname resolution** for **pathnames** (stisystemer) som ikke begynder med slash ("/")



POSIX generelle koncepter

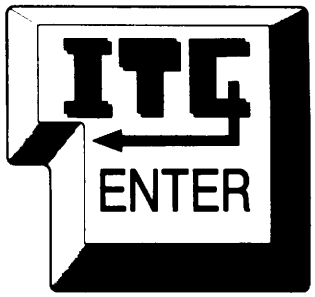
Udvidede sikkerhedskontroller ('extended security controls')

Adgangskontrol ('file access permissions') og privilegie mekanismer blev i standarden defineret på en måde som tillader implementeringen af (implementering-defineret) udvidede sikkerhedskontroller. Disse tillader at forsyne implementeringen med sikkerhedsmekanismer som kan tilgodese forkellige sikkerhedspolitiker end dem som er beskrevet i standarden. Disse implementeringer/mekanismer må dog ikke forandre eller tilsidesætte de semantikker som er definerede i standarden.

Fil-adgangskontrol ('file acces permissions')

Standardens fil-adgangskontrol mekanisme anvender adgangs-bits som beskrevet nedenunder. Disse bits er sat når filen bliver åbnet ved hjælp af *open*(), *creat*(), *mkdir*() og *mkfifo*(), og kan blive ændret ved brug af *chmod*(). Disse bits bliver læst af *stat*() eller *fstat*(). Implementeringen af standarden kan supplere eller tillade andre adgangskontrol mekanismer eller supportere begge typer (standard og udvidet). Kravene til alternativ adgangskontrol mekanisme er følgende, denne skal:

- specificere fil-adgangskontrol for fil ejeren, fil gruppe klassen og alle andre brugere, og som i værdi svarer til den værdi som bliver returneret af *stat*() eller *fstat*()
- kan aktiveres kun af bestemt ejer eller bruger af filen (på per fil basis), ved hjælp af en specificeret aktion
- kan blive koblet fra en fil, efter at fil-adgangskontrol bits (for denne fil) bliver ændret ved hjælp *chmod*(). Frakobling af de alternative mekanismer behøver ikke at frakoble nogle af de andre supplerende mekanismer som er defineret af implementationen. I



alle tilfælde, når processen rekvirerer fil-adgangskontrol, og hvis additionale mekanismer ikke forbyder denne adgang, bliver adgangen bestemt af følgende kriterier:

* hvis processen har de nødvendige privilegier:

    1) hvis læs, skriv eller katalog søgning er krævet, bliver adgangen bevilget

    2) hvis eksekveringstilladelse er krævet, bliver adgangen bevilget til mindst en bruger, af fil-adgangskontrol bits eller af en alternativ mekanisme, ellers bliver adgangen nægtet

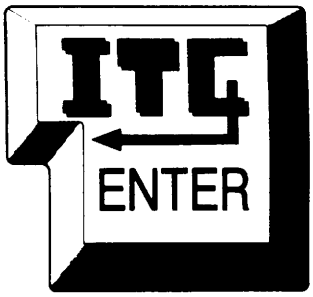
* i andre tilfælde:

    1) fil-adgangskontrol bits for en fil indeholder læse, skrive og eksekvere/søge tilladelse for fil ejeren, fil gruppe klassen eller andre brugere

    2) adgangen bliver bevilget hvis en alternativ adgangskontrol mekanisme er ikke aktiv og de krævede adgangskontrol bits er sat for klassen af processer til hvilken denne process tilhører, eller hvis en alternativ adgangskontrol mekanisme er aktiv og tillader den krævede adgang, ellers bliver adgangen nægtet.

Fil-rangfølge

Filer i et system er organiseret i en rangfølge struktur i hvilken alle ikke terminal-noder er kataloger (directories), og alle terminal-noder er hvilken som helst type af filer. Fordi flere katalog indgange kan pege på samme fil, er rangfølge af filer passende beskrevet som en direkte graf.



Fil-navn portabilitet

Fil-navne bør konstrueres fra portabelt karakter set, fordi brugen af andre karakter set kan være forvirrende eller være flertydig i nogle sammenhænge.

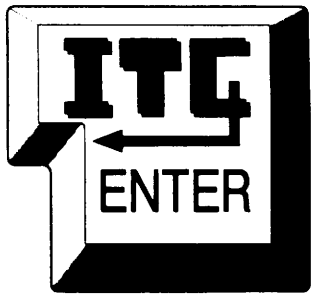
File tids-opdatering

Hver fil indeholder tilknyttede tidsværdier som bliver opdateret når filens data bliver accessed, modificeret eller filens status er blevet ændret. Disse værdier bliver returneret i filens karakteristika-struktur som beskrevet i **<sys/stat.h>**. For hver funktion i POSIX standarden som læser eller skriver, eller ændrer filens status, bliver de relevante tids-relaterede felter markeret som opdaterede. Alle filer som er mærket til opdatering skal opdateres når filen ikke længere er åben af nogen af processer eller når *stat*(), eller *fstat*() bliver udført på filen. Opdateringer bliver ikke udført for filer i **read-only** fil-systemet.

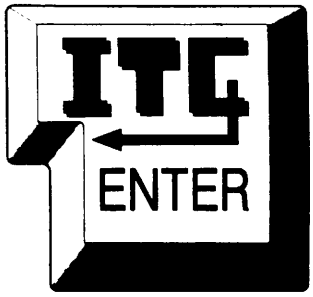
Stisystemets opløsning (pathname resolution)

Stisystemets opløsning udføres af en process for at finde frem til en bestemt fil i fil-rangfølgen, det er mulig at der findes flere stisystemer som finder til samme fil (link).

Hvert filnavn i stisystemet er placeret i kataloget specificeret af katalogets forgænger (for eksempel i stisystemet **a/b**, er **b** placeret i kataloget **a**). Stisystemets opløsning fejler hvis denne ikke kan udføres tilfredstillende. Hvis stinavnet begynder med "/" er forgængeren til dette stisystem **root**'ens katalog (den type af stisystem-opløsning kaldes som absolut stisystem). Hvis stinavnet ikke begynder med "/" er forgængeren til dette stisystem taget fra det øjeblikkelige arbejdskatalog for processen (den type af stisystem-opløsning kaldes som relativt stisystem).



Fortolkningen af stisystem komponenter er afhængig af værdier, af [NAME_MAX] og [_POSIX_NO_TRUNC] associerede med stiens prefix af denne komponent. Hvis en af navne komponenter er længere end [NAME_MAX] og [_POSIX_NO_TRUNC] er effektiv for stiens prefix af denne komponent (jvnfr. *pathconf*()), skal implementeringen opfatte denne som en fejlkondition, ellers skal implementationen bruge den første [NAME_MAX] byte af stiens navne komponenten. Speciel fil-navn **dot**, refererer til det aktuelle katalog i stisystemet og bliver specificeret af forgænger kataloget. Speciel fil-navn **dot-dot**, refererer til forgænger kataloget på niveauet oven over dette katalog. Som specielt tilfælde **dot-dot** i **root**'ens katalog refererer til root'ens katalog som sådan. Stinavn bestående kun af "/" tegnet opløses til at være **root**'ens katalog, af processen. Stinavnet med værdi **null** er ikke gyldig.



Fejlmeddelelser (error numbers)

De fleste funktioner i POSIX standarden sørger for fejlmeddelelser (**error numbers**) i en ekstern variable *errno*, som defineres på følgende måde:

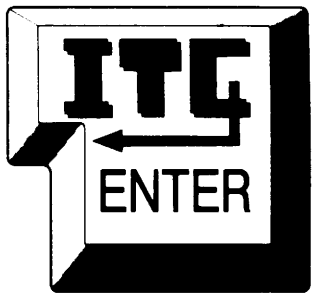
extern int errno;

Værdien af denne variable bør defineres kun efter et kald til en funktion for hvilken (funktion) dette er udtrykkeligt ønsket og sat, og indtil dette er ændret af næste funktion kald. Variablen *errno* bør kun eksamineres i de tilfælde hvor dette er indikeret gyldig af en funktions tilbageværdi (**return value**). Ingen funktioner definerede i POSIX standarden sætter *errno* værdi til **zero (0)** for at indikere fejlkondition.

Hvis der ved eksekveringen af en funktion optræder mere end en fejlkondition, definerer standarden ikke i hvilken rækkefølge disse konditiner skal detekteres, derfor kan en vilkårlig fejlværdi returneres.

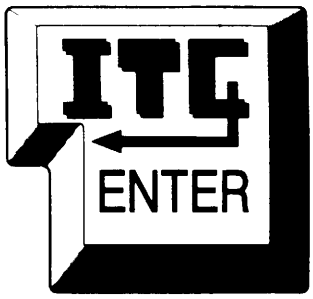
Implementeringer må gerne supportere additionelle fejlværdier som ikke er inkluderet i standarden, under omstændigheder som er anderledes end dem som er beskrevet i standarden eller må gerne indeholde udvidelser, eller begrænsninger som beskytter nogle fejlværdier fra at optræde. Alle funktions beskrivelser i standarden indeholder specifikationer af hvilke fejlomstændigheder kræver fejlmeddelelser og hvilke fejlomstændigheder er implementering-defineret. Implementeringer af standarden må ikke generere fejlværdier som er forskellige fra dem som er beskrevet i standarden.

Følgende symbolske navne identificerer de mulige fejlværdier, i forholdet til de funktioner som er defineret i standarden, disse generelle beskrivelser er mere specifikt defineret i "**Errors**" sektioner af de funktionsdefinitioner som returnerer disse værdier. Kun de symbolske navne bør bruges i programmer siden de aktuelle værdier er implementering-defineret. Alle

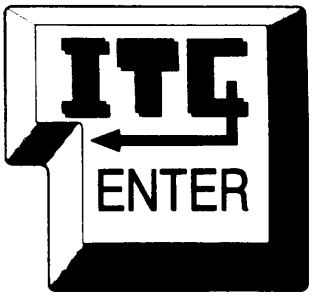


værdier som er listet skal være unike. Implementering-defineret værdier for disse navne skal kunne findes i header-filen **<errno.h>**. Her kommer nogle eksempler på de symbolske navne for fejlmeddelelser:

[E2BIG]	argument liste er for lang
[EACCES]	adgang ikke tilladt (**permission denied**)
[EAGAIN]	resourcen midlertidigt ikke tilgængelig
[EBADF]	dårlig **file descriptor**
[EBUSY]	resourcen er optaget
[ECHILD]	ingen "børne"-proces, en *wait*() eller *waitpid*() funktion blev eksekveret af en proces som ikke har nogle eksisterende "børne"-processer
[EDOM]	domæne fejl, defineret i "C" standarden
[EEXIST]	file eksisterer, eksisterende fil blev refereret på ikke passende måde f.eks. som en ny **link** navn i *link*() funktionen
[EFAULT]	fejl i adresseringen
[EFBIG]	filen er for stort
[EINTR]	afbrydende/forstyrrende funktionskald, et asynkront signal (som f.eks. SIGINT eller SIGQUIT som beskrevet under header-filen **<signal.h>**), blev fanget af en proces som ikke må afbrydes/forstyrres
[EINVAL]	ikke gyldigt (**valid**) argument
[EIO]	Input/Output fejl



[EMLINK]	for mange links
[ENFILE]	for mange åbne filer i systemet
[ENODEV]	**device**'n eksisterer ikke
[ENOENT]	fil eller katalog med det navn eksisterer ikke
[ENOEXEC]	format fejl i eksekverbar fil
[ENOMEM]	ikke nok hukommelse (memory)
[ENOSYS]	den kaldte funktion er ikke implementeret
[ENOTDIR]	det forventede navn er ikke et katalog
[ENXIO]	der findes ikke en sådan device eller adresse
[EPERM]	operationen er ikke tilladt
[EPIPE]	såkaldt **broken pipe**
[ERANGE]	resultatet er for stort, defineret i "C" standarden
[EROFS]	**read-only** fil-system, ingen skrive tilladelser
[ESPIPE]	ikke tilladt søgning (**seek**), en *lseek*() funktion er blevet sat igang overfor en **pipe** eller **FIFO**
[ESRCH]	den specificerede proces findes ikke
[EXDEV]	ikke korrekt udført **link** operation

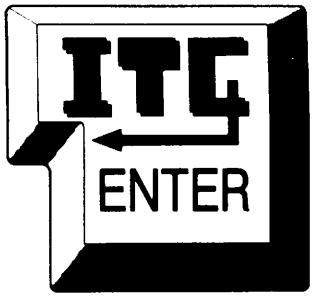


System data typer

Nogle data typer som bliver brugt af forskellige systemfunktioner er ikke defineret som en del af standarden, men bliver defineret af implementationen. Disse typer skal sidenhen defineres i header-filen **<sys/types.h>**, filen skal indeholde i det mindste definitioner på data typer vist i nedenstående tabel:

Defineret type	Beskrivelse af typen
dev_t	brugt for **device** nummer
gid_t	brugt for gruppens ID
ino_t	brugt for filens serienummer
mode_t	brugt til at beskrive fil attributer, f.eks. fil-type, fil-adgangstilladelser og lign.
nlink_t	brugt til at tælle antal link's
off_t	brugt til at bestemme fil-størrelser
pid_t	brugt til at bestemme process ID og gruppe ID
uid_t	brugt til at bestemme bruger ID

Alle de typer som er listede i tabellen skal være af aritmetisk-type, *pid_t* skal være af **signed** aritmetisk-type. Additionelle implementering-definerede definitioner af data typer kan være beskrevet i header-filen, disse definitioner skal have navne som slutter med *_t*.



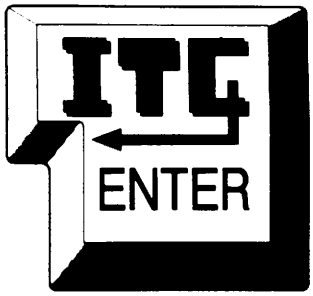
Miljø beskrivelse (environment)

En række (**array**) af strenge kald miljø (**environment**) er tilgængelige når processen starter, denne række er pointeret til at være ekstern variable *environ*, som bliver defineret på følgende måde:

extern char **environ;

Disse strenge har følgende form *navn=værdi*, hvor *navn* ikke må indeholde karakteren "=". Det er underordnet i hvilken rækkefølge strengene fremkommer i miljøet. Hvis mere end en streng i proces miljøet har det samme "navn" er konsekvencen af dette ikke defineret. Følgende "navne" kan blive defineret og i så fald har de mening som beskrevet nedenfor:

HOME	navnet på brugerens initiale arbejds-katalog, taget fra bruger database (som beskrevet i header-filen **<pwd.h>**)
LANG	navnet for foruddefineret opsætning af lokal miljø
LC_COLLATE	navnet for lokal sorteringssekvens
LC_MONETARY	navnet for lokal monetær, numerisk information
LC_NUMMERIC	navnet for lokal editering af numeriske værdier (f.eks. separator komma og/eller punktum)
LC_TIME	navnet for lokal formattering af dato/tid informationer
LOGNAME	navn for brugerens **login** konto, svarende til brugerens login-navn og taget fra bruger database (som beskrevet i header-filen **<pwd.h>**)

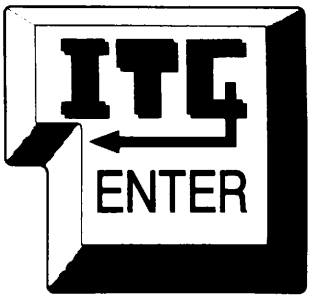


PATH	en sekvens af sti-præfikser, således at søgning efter en fil som kun kendes ved navn er mulig, uden at man kender det fulde sti-system
TERM	beskrivelse af terminal-karakteristika med hensyn til formatering af output data
TZ	tids-zone informationer

Miljø-variable "navne" brugt eller fremstillet af en applikation burde kun bestå af karakterer fra såkaldt "portabelt fil-navn karakter set", andre karakterer kan være tilladte af en applikation og applikationer burde tolerere eksistensen af disse navne.

"værdier" som man kan tilegne miljø-variable indeholder ingen restriktioner, bortset fra at de skal afsluttes med en **null** byte og den totale plads som bruges for at opbevare disse argumenter er limiteret til værdi af [ARG_MAX] bytes.

Andre *navn=værdi* kombinationer kan placeres i miljø beskrivelsen ved hjælp af *environ* variablen eller ved brug af *envp* argumenter når en proces startes.



POSIX symboler

Nogle symboler i POSIX standarden er defineret i header-filer. Nogle af disse header-filer kunne også definere andre symboler end dem som standard definerer, der eksisterer potentielle konflikt muligheder med symboler som bliver brugt af applikationer, kort sagt POSIX standarden definerer de symboler som ikke må bruges af andre standarder i deres header-filer uden at der eksisterer en vis kontrol omkring synligheden af disse symboler.

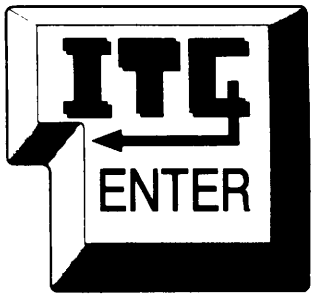
Symboler som kaldes **feature test macros** er brugt til at kontrollere synligheden af de symboler som måtte blive inkluderet i header-filer. Implementeringer af POSIX standarden, fremtidige versioner af standarden og andre standarder kan definere yderligere **feature test macros. feature test macros** burde defineres i kompileringen af en applikation før en `#include` af nogen header-fil hvor symbolerne skal gøres synlige for nogle applikationer men ikke for dem allesammen. Hvis definitionen af macroen ikke forekommer før `#include`, bliver resultatet udefineret.

Faciliteter i test macroen skal begynde med understregningstegnet ("_").

Følgende **feature test macro** er defineret i POSIX standarden:

_POSIX_SOURCE betyder at programmer forventer, at symboler defineret af POSIX standarden bliver forsynet fra miljøet. Hvor udvidelser er tilladte i header-filen, men ingen udtrykkelig tvang af formen af navnet er supporteret, af standarden, skal disse udvidelser ikke gøres synlige ved brugen af **feature test macro.**

Den eksakte betydning af denne feature er afhængig af denne type af "C" programmeringssproget support som man vælger ved implementeringen.



POSIX standarden og programmeringssproget "C"

Nogle termer og symboler som bliver brugt i POSIX standarden anses at blive defineret i kommende "C" standard. Blandt andet følgende termer defineres af "C" standarden:

CLK_TCK
NULL
byte

character array
clock_t
header

null character

string

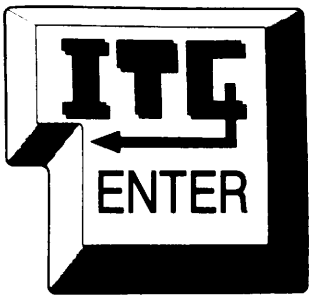
time_t

Termen **NULL pointer** som bliver brugt i POSIX standarden er lig termen **null pointer** som brugt i "C" standarden. Symbolet **NULL** skal erklæres i **<unistd.h>**, med samme værdi som krævet af "C" standarden og i forlængelse af flere forskellige lokationer som allerede krævet af "C" standarden.

Derudover gør følgende sig gældende med hensyn til symboler som begynder med understregningstegn ("_"):

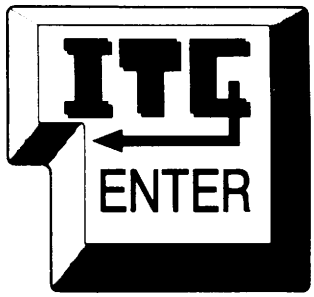
* alle eksterne identifikatorer (**identifiers**) som begynder med understregningstegn "_" er reserverede

* alle andre identifikatorer som begynder med understregningstegn og enten et stort bogstav (**upper case**) eller et andet understregningstegn er reserverede



* om programmet definerer en ekstern identifikator med det samme navn som en reserveret ekstern identifikator, selvom dennes semantik har samme form, er opførslen af denne ikke defineret

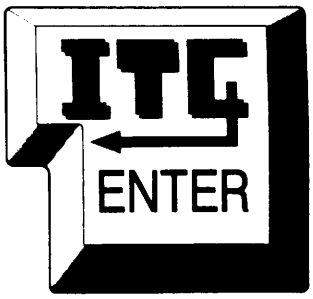
Nogle andre navneområder er reserverede af "C" standarden, disse reservationer gør sig gældende også overfor POSIX standarden. Desforuden kræver "C" standarden at der skal være mulighed for at inkludere en header mere end een gang og at symboler kan defineres i flere end een header, dette krav er også gældende med hensyn til headers i POSIX standarden.



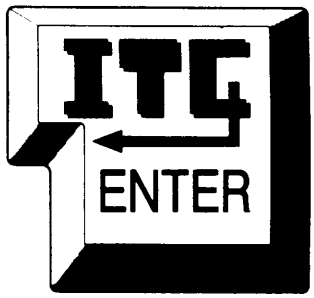
Headers og funktionsprototyper

Implementeringer som kræver "C" standard sprog-afhængig support skal erklære funktionsprototyper for alle funktioner. Implementeringer som kræver fælles brug af "C" sprog-afhængig support skal erklære resultat typer for alle funktioner som ikke returnerer en "klar" *int*. Disse funktioners prototyper (om nødvendig) skal fremkomme i headers som er listet i nedenstående tabel. Hvis funktionen ikke er listet i tabellen, så skal denne funktions prototype fremkomme i **<unistd.h>**, som antages at blive `#include`'ret når hvilken som helst funktion som er erklæret i denne bliver brugt. Kravene til synligheden af symboler i POSIX standarden skal honoreres.

<dirent.h>	*opendir*(), *readdir*(), *rewinddir*() og *closedir*()
<fcntl.h>	*open*(), *creat*() og *fcntl*()
<grp.h>	*getgrgid*() og *getgrnam*()
<pwd.h>	*getpwuid*() og *getpwnam*()
<setjmp.h>	*sigsetjmp*() og *siglongjmp*()
<signal.h>	*kill*(), *sigemptyset*(), *sigfillset*(), *sigaddset*(), *sigdelset*(), *sigismember*(), *sigaction*(), *sigprocmask*(), *sigpending*() og *sigsuspend*()
<stdio.h>	*fileno*() og *fdopen*()
<sys/stat.h>	*umask*(), *mkdir*(), *mkfifo*(), *stat*(), *fstat*() og *chmod*()
<sys/times.h>	*times*()
<sys/utsname.h>	*uname*()



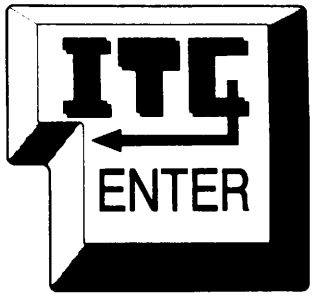
<sys/wait.h>	*wait*() og *waitpid*()
<termios.h>	*cfgetospeed*(), *cfsetospeed*(), *cfgetispeed*(), *cfsetispeed*(), *tcgetattr*(), *tcsetattr*(), *tcsendbreak*(), *tcdrain*(), *tcflush*() og *tcflow*()
<time.h>	*time*() og *tzset*()
<utime.h>	*utime*()



Numeriske begrænsninger

Følgende afsnit beskriver størrelsen af begrænsninger som er blevet påtvunget af specifike implementeringer. Notationen [LIMIT] er brugt i standarden for at indikere disse værdier, men selve udtrykket [LIMIT] er ikke en del af standarden. Nogle af begrænsningerne forventes at blive defineret i "C" standarden. På nuværende tidspunkt er følgende begrænsninger defineret i "C" standarden:

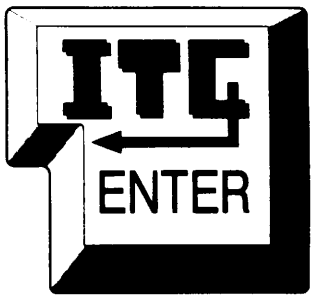
[CHAR_BIT],
[CHAR_MAX],
[CHAR_MIN],
[INT_MAX],
[INT_MIN],
[LONG_MAX],
[LONG_MIN],
[MB_LEN_MAX],
[SCHAR_MAX],
[SCHAR_MIN],
[SHRT_MAX],
[SHRT_MIN],
[UCHAR_MAX],
[UINT_MAX],
[ULONG_MAX] og
[USHRT_MAX].



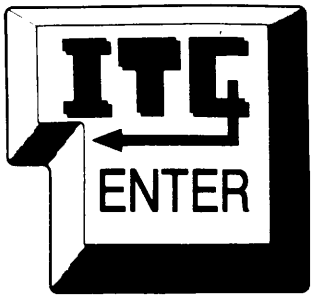
Minimale værdier

Symboler i nedenstående tabel skal være defineret i **<limits.h>** med værdier som vist i tabellen. Disse er symbolske navne for de mest restriktive værdier for sikre karakteristika, for et system som er konform med POSIX standarden. De relaterede symboler er beskrevet i standarden og behøver ikke at være så meget restriktive. En konform implementering skal forsyne værdier som i det mindste er af denne størrelse. Portable applikationer skal ikke gøre krav på større værdier for at kunne udføres korrekt.

Navn	Beskrivelse	Værdi
[_POSIX_ARG_MAX]	længden af argumentet for een af `exec` funktioner i bytes, inklusive miljø data	4096
[_POSIX_CHILD_MAX]	nummer af samtidige processer per een reel bruger ID	6
[_POSIX_LINK_MAX]	værdi af filernes link tæller	8
[_POSIX_MAX_CANON]	nummer af bytes i terminalens forskriftsmæssige input-kø	255
[_POSIX_MAX_INPUT]	nummer af bytes for hvilke det er reserveret område i terminalens input-kø	255
[_POSIX_NAME_MAX]	nummer af bytes i en fil-navn	14



[_POSIX_NGROUPS_MAX]	0
nummer af samtidige suplementære gruppe ID'er per process	
[_POSIX_OPEN_MAX]	16
nummer af filer som en process kan have åbnet samtidigt	
[_POSIX_PATH_MAX]	255
nummer af bytes i et stinavn inklusive evt. fil-navn	
[_POSIX_PIPE_BUF]	512
nummer af bytes som kan skrives **atomisk** til en `pipe`	



Symbolske konstanter

En konform implementering af standarden skal indeholde en header-fil **<unistd.h>**, denne fil definerer symbolske konstanter og strukturer som er definerede i standarden. Konstanter defineret af denne header-fil er vist nedenfor, de aktuelle værdier af disse konstanter er implementering-defineret.

* Symbolske konstanter for *access*() funktionen.

Konstanten	Beskrivelsen
R_OK	test for læsetilladelse
W_OK	test for skrivetilladelse
X_OK	test for eksekverings- eller søgetilladelse
F_OK	test for at en fil eksisterer

Konstanter F_OK, R_OK, W_OK og X_OK og udtrykkene:

R_OK | W_OK

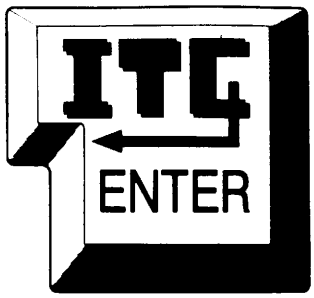
R_OK | X_OK

R_OK | W_OK | X_OK

hvor tegnet "|" representerer **bitwise** inkluderet **OR** operator, skal alle have forskellige værdier.

** Symbolske konstanter for *lseek*() funktionen

Konstanten	Beskrivelsen
SEEK_SET	set fil-offset til *offset*
SEEK_CUR	set fil-offset til aktuel værdi plus *offset*



SEEK_END	set fil-offset til EOF mærket plus *offset*
Compile-Time	symbolske konstanter for portabilitetsspecifikationer.

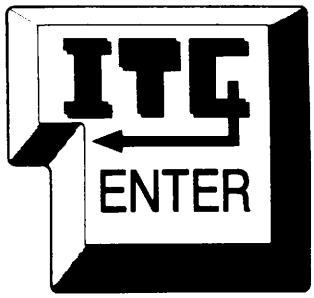
Konstanter som er listet nedenunder kan blive brugt af en applikation i kompileringsøjeblikket for at fastslå hvilke optionale faciliteter der findes på systemet og hvilke aktioner implementationen ville tage.

Compile-Time symbolske konstanter.

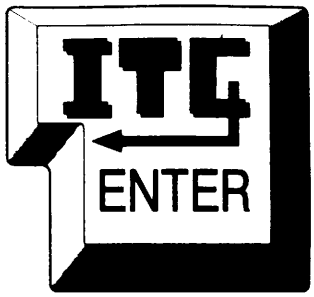
Navn	Beskrivelse
[_POSIX_JOB_CONTROL]	hvis dette symbol er defineret, dette indikerer at implementeringen supporterer job kontrollen
[_POSIX_SAVED_IDS]	hvis defineret, hver process har gemt **set-user-ID** og **set-group-ID**
[_POSIX_VERSION]	heltal (integer) værdi 198808, denne værdi vil blive ændret med alle publicrede versioner eller revisioner af POSIX standarden for at indikere (4-digit) år og (2-digit) måned, af godkendelsen

Execution-Time symbolske konstanter for portabilitetsspecifikationer.

Disse konstanter kan blive brugt af applikationen på eksekveringstidspunktet for at fastslå hvilke optionale faciliteter er implementeret på systemet og hvilken aktion tages af implementationen i de tilfælde som er beskrevet i standarden som implementering-defineret.



Navn	Beskrivelse
[_POSIX_CHOWN_RESTRICTED]	brug af *chown*() funktionen er begrænset til en process med de rigtige privilegier
[_POSIX_NO_TRUNC]	stinavn komponenter som er længere end [NAME_MAX] genererer en fejlkondition
[_POSIX_VDISABLE]	terminal specifikke karakterer definerede i **<termios.h>** kan blive frakoblet ved brug af denne karakter-værdi, hvis denne er defineret (se *tcgetattr*() og *tcsetattr*()

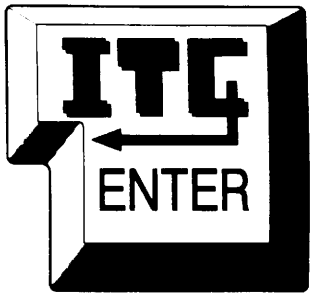


POSIX standarden og "C" programmeringssprog standarden

Som nævnt i denne guide er nuværende forslag til POSIX standarden baseret på UNIX systemets miljø og meget tæt knyttet til kommende standard med hensyn til sproget "C".

I skrivende stund findes der ikke en international version, godkendt af ISO's medlemslande, af "C" standarden. Derfor peger alle henvisninger i den foreslåede POSIX standard på den **Draft Proposal** som findes i dag i "C" arbejdsgruppen (ISO/IEC JTC 1/SC 22/WG 14) og som er dateret maj 1988.

POSIX standarden har dog forpligtet sig til at følge de eventuelle ændringer i sproget, som måtte komme, som følge af den registrering og internationale afstemning, som vil finde sted i forbindelse med vedtagelsen af "C" standarden.



Afsluttende bemærkninger

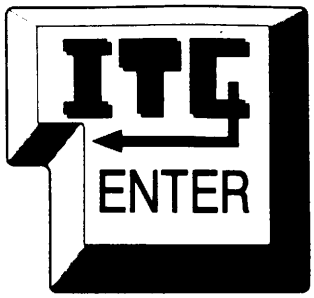
Denne guide beskriver hovedsagelig POSIX standardens definition af kernel (operativ systemets) interface til andre dele af systemet. Dette sker på baggrund af at det er kun første del af standarden som er blevet registreret på nuværende tidspunkt.

Hvis afstemninger blandt medlemsnationer af ISO forløber uden større problemer og der ikke kommer mange nye ting som man mener bør inkluderes i denne standard så skulle den del af standarden blive vedtaget som international standard (IS) omkring midten af 1989.

Med hensyn til de resterende dele af standarden så tager det nok noget længere tid. Den del som er længst fremme med hensyn til at fremstå som et færdigt dokument som eventuelt kan registreres af ISO som "Draft Proposal" er delen som omfatter standarden med hensyn til beskrivelser af "shell" funktioner og "utilities" for POSIX's set af interfacer. Vi i Danmark regner med at denne del af standarden kan blive registreret som DP i løbet af første halvår af 1989. Dette vil indebære at eventuelt registrering af denne del som "Draft International Standard" (DIS), vil kunne ske i slutningen af 1989, og derfor, hvis alt vil gå planmæssigt, så kan man eventuelt forvente at denne del foreligger som International Standard i løbet af 1990, hen imod slutningen af året.

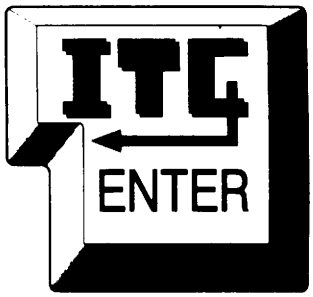
Med hensyn til de resterende dele af standarden, som er blevet nævnt i indledningen til denne giude, så er det meget svært at gisne om hvornår disse er "modne" til at blive gjort til Internationale Standarder. Een ting er sikkert, at det er mange aktiviteter i POSIX standardiseringsgruppen både på det hjemlige plan og på det internationale.

God fornøjelse med læsningen af standarden !!!



Dansk POSIX guide

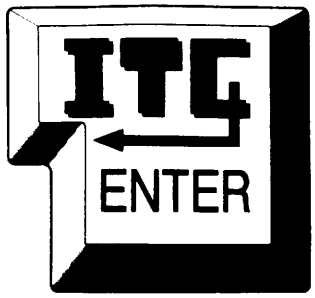
Tillægssektion



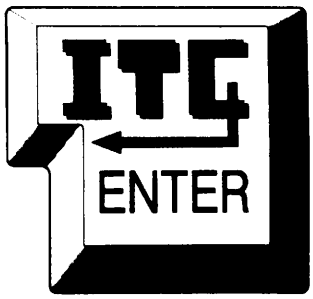
Tillæg 1

Forkortelsesliste

A	Arbejdsgruppe i et underudvalg under DS
ANSI	American National Standards Institute
CCITT	Comite Consultatif International Telegraphique et Telephonique
CEN	Commission Europeenne de Normalisation
CENELEC	Commission Europeenne de Normalisation ELECtronique
CGM	Computer Graphics Metafile
CUT	Coordinated Universal Time
DFS	Distributed File System
DIS	Draft International Standard
DP	Draft Proposal
DS	Dansk Standardiseringsråd
EDI	Electronic Data Interchange
EN	European Norm
ENV	European Trial (Versuch) Norm
FIMS	Forms Interface Management System
FIPS	Federal Information Processing Standards
FTAM	File Transfer And Manipulation
GKS	Graphical Kernel System
IEC	International Electrotechnical Committee
IRDS	Information Resource Dictionary Service
IS	International Standard
ISO	International Standardisation Organisation
IT	Information Technology



JTC1	ISO/IEC Join Technical Committee 1
LAN	Local Area Network
MHS	Message Handling System
NDL	Network Database Language
NFS	Network File System
NIST	National Institute for Standards and Technology (tidligere NBS – National Bureau of Standards)
NWI	New Working Item
ODA	Office Document Architecture
ODIF	Office Document Intechange Format
OSCRL	Operating System Control and Request Language
OSI	Open System Interconnection
PHIGS	Programmers Hierarchical Interactive Graphics System
POSIX	Portable Operating System Interface for Computer Environment
RFS	Remote File System
S	Hovedudvalg under DS
SC	SubCommittee
SQL	Structured Query Language
u	Underudvalg under hovedudvalg, under DS
UPE	User Programming Environment
WAN	Wide Area Network
WD	Working Draft
WG	Working Group



Tillæg 2

Litteraturfortegnelse

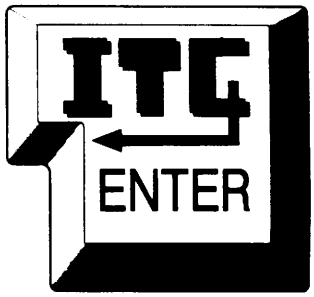
1. "ISO/IEC JTC1/SC22 N585 Final Version of DP9945- Portable Operating System Interface for Computer Environments." ISO – November 1988

2. "AT&T, UNIX Time Sharing System: UNIX Programmer's Manual, Seventh Edition" – Bell Telephone Laboratories, Inc. – Januar 1979

3. "AT&T, UNIX System III Programmer's Manual" Western Electric Company, Inc. – Oktober 1981

4. "AT&T, System V Interface Definition, Issue 2" AT&T – 1986

5. "4.3 Berkeley Software Distribution, Virtual VAX-11 Version" The Regents of the University of California, Berkeley – April 1986



Dansk POSIX guide

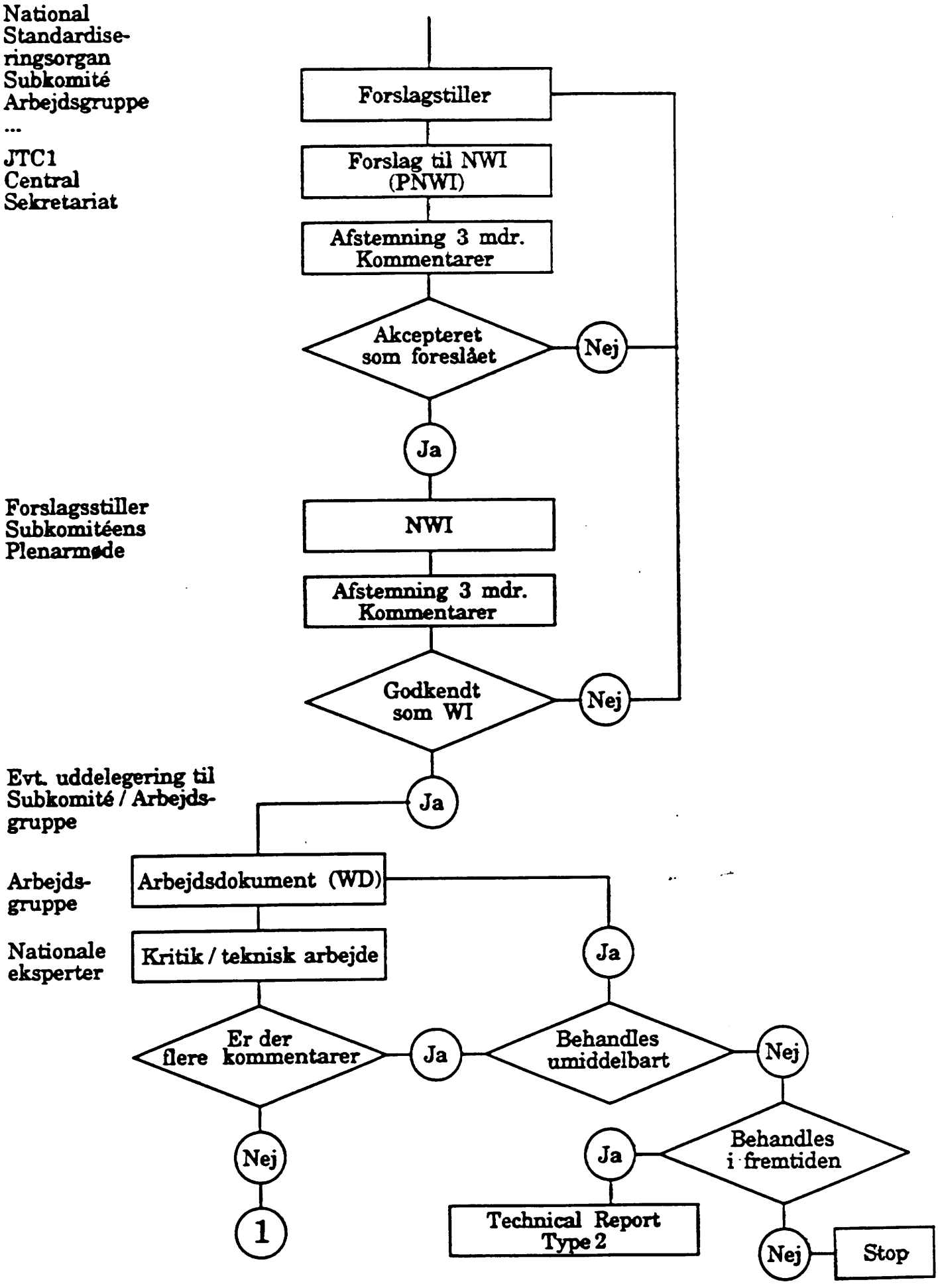
Tillæg 3

Tilblivelse af standarder

Stadier af teknisk arbejde

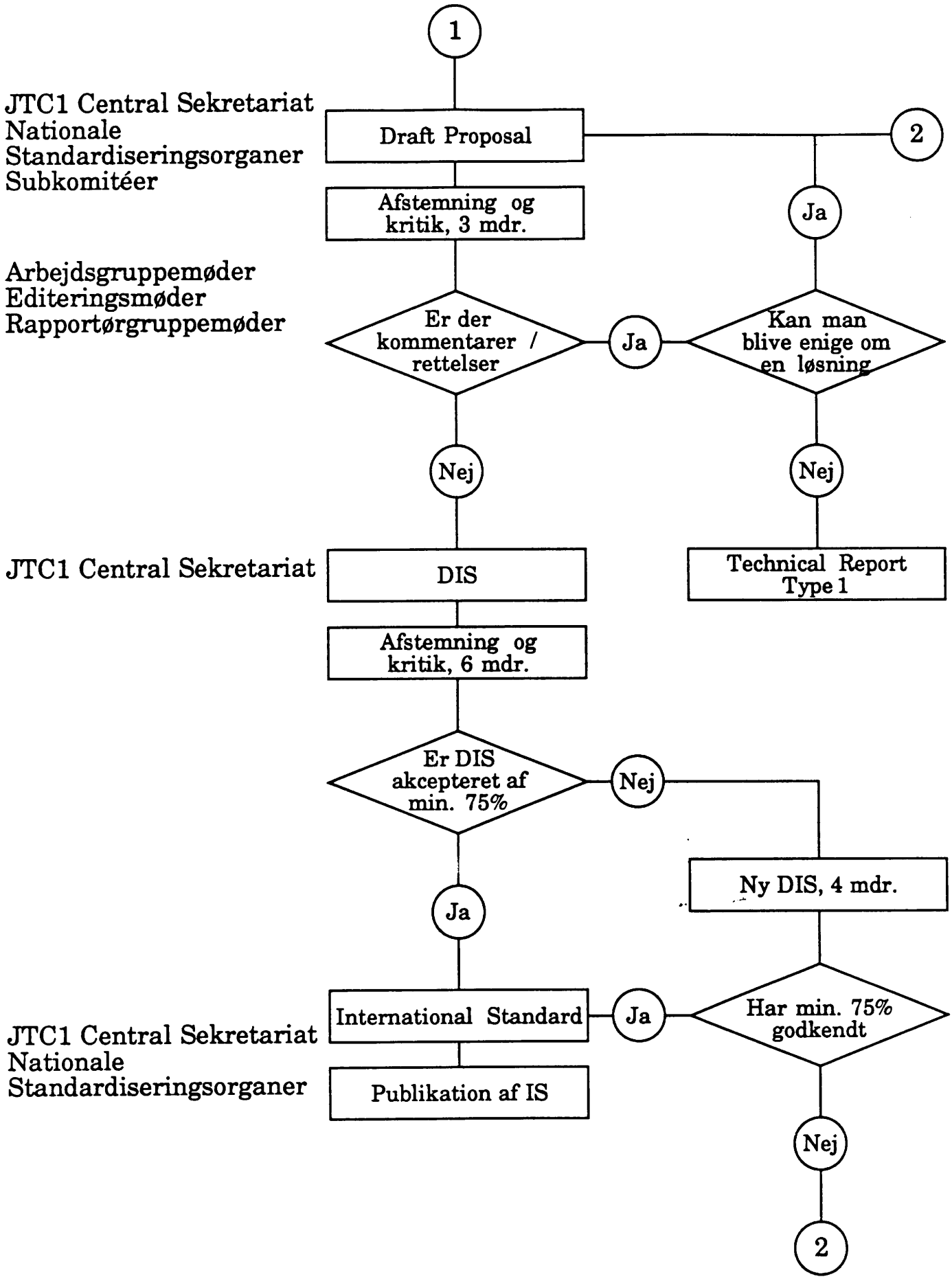
Stadie 1: (forslagsfase)	Forslag til ny standardiseringsarbejde (New Working Item – NWI), under overvejelse.
Stadie 2: (forberedelsesfase)	Arbejdsdokumentet er under forberedelse (Working Draft – WD)
Stadie 3: (komitéfase)	Udkast til foreslået standard er under overvejelse (Draft Proposal – DP)
Stadie 4: (afstemningsfase)	Arbejdsdokumentet er stabilt og sendt til afstemning som foreslået standard (Draft International Standard – DIS)
Stadie 5: (publiceringsfase)	International Standard er forberedt til publicering (International Standard – IS)

Tilblivelse af standarder – Flow diagram



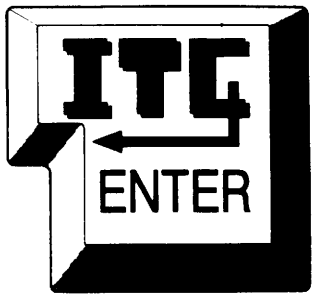
Isak Kom, IT Center

Tilblivelse af standarder – Flow diagram



Revisionsfrist: senest 5 år efter publikation

Isak Korn, IT Center



Dansk POSIX guide

Tillæg 4

COMMISSION
OF THE
EUROPEAN COMMUNITIES

DG XIII
Telecommunications
Information Industries
and Innovation

Brussels, 06.09.1988

Original Lang.:

Working Document
Nr. 260

SOGITS

Senior Officials Group
Information Technologies Standardization

Title: BC-IT-102: POSIX

Original Title:

Chairman:

Michel CARPENTIER
C. E. C.
DG XIII, RJ II-70
Rue de la Loi 200

B - 1049 Brussels

Tel.: 235.95.49

Secretary

Eckhard Koch
C. E. C.
DG XIII, RJ II-37
Rue de la loi 200

B - 1049 Brussels

Tel.: 235.97.02

Brussels, 31 August 1988
Programme IT
Dir. 83/189/CEE
BCIT-102E

D R A F T

Standardization order forwarded to CEN/CENELEC in the framework of the standardization work done by CEN/CENELEC with the cooperation of the CEPT in the fields of Information Technology and Telecommunications

PURPOSE

Drafting of a set of European Standards (ENV in the first phase) on the basis of the agreed standardization programme in the framework of a common application environment based on POSIX.

JUSTIFICATION

In many data processing applications, the exchange of data and information depend not only upon the basic data communication standards but also upon the computer operating systems.

Most of the current Operating Systems are proprietary products and the transfer of application software outside this specific environment is practically impossible without complex adaptation and high training costs. The concept of Open Software has led to the definition of a single Operating System and particularly GPOS (Generalized and Portable Operating System), which can also be run on different machines from various suppliers.

The advent of UNIX, a GPOS written in a high level language (C) and the licencing policy of ATT, owner of UNIX, have helped to make the product available from the most important computers suppliers. UNIX market grew quickly but, at the same time, different not fully compatible variants were offered by industry.

In 1987, ISO TC 97 SC 22 adopted the IEEE P1003.1 standard (POSIX), a derivative of UNIX, as the base for the future ISO Standard. The standard "portable operating system" corresponds to the needs expressed by many manufacturers and users and the requirements for an harmonized solution in this area have been already taken up in the latest CEN programme.

ORDER

Draft a set of European Standards (ENV) corresponding to the POSIX interface based on the agreed CEN programme (SOGITS N 231).

The resulting Standards will be drafted with enough accurary to provide the adequate basis for ensuring conformity testing.

DEADLINES

1. Preparation of the draft European Standard	06.89
2. Adoption of the draft as European Standard	01.90
3. Transposition of the European Standard in accordance with the procedures applicable to EN Standards	04.90

ALIGNMENT TO INTERNATIONAL STANDARDS

When drafting the requested Standards, the standardization organizations should seek for maximum alignment with the results of ongoing work in ISO TC 97 SC 22 (DP 9945)).

STANDSTILL

The standstill applies for the Standards developed under this order voucher (Article 7 of Directive 83/189).

PUBLICATION OF THE STANDARD IN THE OFFICIAL JOURNAL

Title and summary.

NOTE

1. The CEN/CENELEC General Secretariat will inform the Commission as soon as the degree of stability of the ENV standard becomes sufficient to draft a standardization order allowing its final adoption as an EN standard (within a maximum of three years).

2. It would be desirable that draft standards and European Standards of the ENV-type to be prepared and transmitted electronically and made accessible through the existing public telecommunications services in accordance with International Standards, as well as in the form of a normal A4 hard copy.