Herinneringen aan het voor-candidaatspracticum Natuurkunde, 1973 – 1979

Auteurs : Harm Braams, Edgar Hildering, Tom Hoogenboom, Wim Nelis, Fred Rabouw
Versie: 5

0. Inleiding

Aan het begin van het academisch jaar 1973 – 1974 was in het toenmalige Transistorium I een terminalkamer ingericht. Het doel was dat studenten een betere toegang zouden krijgen tot de centrale computer van de Universiteit van Utrecht en daar, ook tijdens het practicum, gebruik van zouden gaan maken. In de betreffende ruimte stonden drie of vier stuks Teletype ASR-33, die tien tekens per seconde konden sturen of ontvangen (110 [b/s], 110 [baud]). Gaandeweg zijn ze vervangen door snellere exemplaren, die 30 tekens per seconde (300 [b/s], 300 [baud]) konden verwerken.

De centrale computer, opgesteld in het “Academisch Computer Centrum Utrecht” (ACCU), was een Control Data Corporation (CDC) Cyber 73-28. De machine was uitgerust met twee stuks “Central Processor Unit” (CPU’s). Als ik het goed onthouden heb betekent de ‘8’ in het modelnummer dat het 8 stuks “Peripheral Processor Unit” (PP’s) bevatte. De PP’s verzorgden de invoer en de uitvoer, met uitzondering van PP0 en PP1: PP0 bestuurde de machine (MTR) en PP1 stuurde de “operator console” aan (DSD). De Cyber 73-28 bevatte 131 [kWord] werkgeheugen, waarin elk woord 60 bits besloeg. (Merk op dat 131 k de marketing variant is van 128 Ki.) De tijd benodigd om één woord te lezen of te schrijven bedroeg 1 [µs]. Dit werkgeheugen werd gedeeld door de twee CPU’s. Elke PP had een eigen werkgeheugen van 4 [KiWord], waarbij een woord 12 bits lang was. Het werkgeheugen was opgebouwd rond ferrietkralen.

Aan het begin van het academische jaar 1973 was het technische gedeelte voor een computerondersteuning bij het practicum klaar, maar er moest nog de nodige software ontwikkeld worden, zoals een programma voor elke daartoe in aanmerking komende proef. Het toeval wil dat in dat jaar tevens een paar studenten aan hun studie natuurkunde begonnen, die tevens zich sterk aangetrokken voelden tot het programmeren van computers. Ter illustratie: Tom Hoogenboom en Wim Nelis kenden elkaar al van de middelbare school. De vader van Tom was professor aan de universiteit, en mogelijk daardoor wist Tom dat er een cursus programmeren gegeven zou worden, nog voor de colleges van het jaar begonnen waren. Dit was een cursus over Algol-60, gegeven door professor van de Sluis. Later (*wanneer?*) hebben we ook een cursus over de machinetaal van de Cyber 73 gevolgd, met als docent professor Veltman.

> Professor Veltman heeft een programma geschreven voor het uitvoeren van analytische mathematische bewerkingen, vooral voor gebruik in hoge-energie fysica. Dit programma heette “schoonschip” (zodat het moeilijk uit te spreken was voor Engelstalige collega’s) en was voor zover ik weet deels geschreven in machinetaal.

Al gauw ontstond er een klein clubje van studenten dat zich op de programmeerwerkzaamheden stortte en ‘en passant’ ook nog andere leuke dingen heeft gedaan.

In de volgende hoofdstukken staan wat herinneringen, niet in chronologische volgorde, aan die tijd waarin we vele uren doorbrachten werkend aan programma’s.

1. Een april

1.0. Aanzet

De kennis over de computer en over het gebruikte OS (SCOPE geheten) waren al redelijk ontwikkeld toen een van de leden van de groep een octale geheugendump bekeek die gemaakt was nadat een programma in ontwikkeling was gecrasht. Bij toeval vond hij de naam van een systeembestand, waarin het bericht-van-de-dag staat opgeslagen. Dit bericht werd op het scherm of in de papieren uitvoer geplaatst. In NOS/BE kon je 4 of 5 wachtwoorden per bestand definiëren, voor verschillende typen toegang tot het bestand. In de geheugendump stonden alle wachtwoorden van het bestand vermeld.

> Als een programma op een legale wijze toegang had gekregen tot een bestand en stopte, werd het werkgeheugen niet gewist. Als een ander programma werkgeheugen aanvroeg kon daarin nog informatie staan die gebruikt was door het vorige programma. Als het hele werkgeheugen vanwege een crash op papier wordt gezet, heb je een kans gevoelige informatie van een vorig programma aan te treffen in die dump. Er is ooit een programma door een groepslid geschreven die dit effect gebruikte: vraag werkgeheugen aan, doorzoek deze op gevoelige informatie, geef het werkgeheugen vrij en wacht enige tijd. Daarna kon de cyclus herhaald worden. Het zou kunnen dat de voornoemde wachtwoorden op deze wijze gevonden zijn.

Met deze informatie werd het mogelijk om zelf een, wat minder geautoriseerde, bericht-van-de-dag te publiceren. Daar het bijna 1 april was, lag een toepassing voor de hand. Er is een bericht samengesteld met als strekking dat het energiegebruik en het papiergebruik door het ACCU teruggedrongen moesten worden. Daarom zou de Cyber 73-28 vervangen worden door een PDP 8, die nog ergens in een hoekje in een gang van het mathematisch centrum stond, en zou papieren uitvoer via de regeldrukkers gestopt worden tot de bomen rond het ACCU voldoende waren gegroeid om daar papier van te maken. Het berichtje had een omvang van ongeveer 30 regels.

Het was de gewoonte van het ACCU om een dergelijk bericht te plaatsen in een kader. Daardoor zag een bericht er ongeveer als volgt uit:

```
ACCUACCUACCUACCUACCUACCUACCUACCUACCUACCUACCUACCU
AC                                              CU
AC  Een testberichtje.                          CU
AC                                              CU
ACCUACCUACCUACCUACCUACCUACCUACCUACCUACCUACCUACCU
```

Dit is natuurlijk netjes over genomen. Een essentieel detail is de betekenis van de eerste letter op een regel uitvoer: deze was in gebruik om de positionering te bepalen en werd in principe niet afgedrukt. Die eerste letter werd de ‘format effector’ genoemd. In de onderstaande tabel staan een beperkt aantal van de mogelijke waarden en hun betekenis vermeld.

Teken	Effect
Spatie	Ga naar de volgende regel en druk de rest van de regel uitvoer af.
+	Druk de rest van de regel uitvoer af. Deze wordt dus over de voorgaande regel afgedrukt.
1	Ga naar het begin van de volgende pagina en druk de rest van de regel af.
A	Druk de rest van de regel af en ga naar het begin van de volgende pagina.

Helaas was er een foutje gemaakt in het geprepareerde bericht: de spatie aan het begin van elke regel ontbrak. Dus elke regel had als ‘format effector’ letter ‘A’.

Op 1 april 1974 *(?)* is een stapeltje ponskaarten (een ‘job’) ingelezen met daarin de opdrachten om het bericht-van-de-dag bestand te vervangen. Het effect was direct merkbaar. Vanuit de wachtruimte van het ACCU had je zicht op de twee regeldrukkers. Deze begonnen al snel met het ‘spuwen’ van vrijwel lege bladzijden. Door de letter ‘A’ op de verkeerde positie begon elk stuk uitvoer met ongeveer 30 bladzijden, elk met slechts één regel tekst. Al gauw werd door medewerkers van het ACCU de verdere behandeling van ‘jobs’ gestopt, het betreffende bestand hersteld, waarna de productie verder kon gaan.

Op een dergelijk gedrag was geanticipeerd. Daarom was dezelfde ‘job’ twee keer ingevoerd, alleen in het tweede exemplaar werd veel werkgeheugen en veel rekentijd (die geen van beiden nodig waren) aangevraagd. Daarom duurde het enige tijd voor deze werd behandeld. Enige tijd nadat het ‘spuwen’ was gestopt, begon het dus weer.

1.1. Vervolg

Een jaar later is de groep te verstaan gegeven dat ze zich op 1 april niet in de buurt van de computer of enige van de aangesloten terminals mochten begeven. Daar hebben we ons bijna aan gehouden.

Een ponsband is een (lange) strook papier waarop dwars op de langsrichting 8 posities beschikbaar zijn om een gaatje te maken. Je kunt dus ook leesbare tekst in zo’n band ponsen. Voor elk teken is een matrix van 8 bij 8 posities beschikbaar. Op een terminal met een ponsbandponser is een aantal bandjes gemaakt, waarop in grote letters de tekst ‘OPERATOR DROP’ geschreven stond. Deze tekst werd in de uitvoer van een ‘job’ geplaatst als de operator van de Cyber om een of andere reden het nodig vond om de betreffende ‘job’ voortijdig te beëindigen.

Tevens zijn een aantal zakken drop gekocht en is aan elke een bandje met de tekst ‘OPERATOR DROP’ geniet. Na vriendelijk vragen mochten we toch het ACCU in en hebben we aan de medewerkers een zakje drop uitgedeeld, te beginnen bij de operator(s) die op dat moment dienst hadden.

1.2. Aanvulling van Harm Braams

Rudi van Houten, die op de derde verdieping van het wiskundegebouw zat, beheerde "de handleiding" van de Cyber. Dat was 150 cm aan boekwerk. Als student was ik obstinaat: handleidingen, dat ga je niet lezen, gewoon proberen en er zelf achter komen. Maar Rudi lachte dan fijntjes en zei: “Even zoeken en je hebt je antwoord”. En inderdaad: het was helemaal niet nodig om op de moeilijke manier wachtwoorden in het geheugen te zoeken, want in de handleiding stond dat je het óók met een universeel wachtwoord kon doen, namelijk "UNIVRPERM". De gouden regel RTFM ("read the fucking manual") heb ik mijn hele leven gebezigd (dank Rudi)!

2. LOS en BB

Er zijn twee serieuze programma’s ontwikkeld voor het practicum. Het betreft LOS, dat programma’s uit een bibliotheek kon laden en starten en tevens statistieken over het gebruik bij hield, en BB, dat meerdere kleine bestandjes van studenten (en stafleden) opgeslagen in een groot bestand beheerde.

2.0. Rationale

Diskruimte op “de” computer was toentertijd een schaars goed. Het voorcandidaatspracticum Natuurkunde had een beperkte hoeveelheid diskruimte toegewezen gekregen, waarop de studenten

hun bestanden, nodig voor de verwerking van de gegevens uit proefjes, konden opslaan. Echter de minimale grootte van een bestand, eigenlijk de kleinste hoeveelheid diskruimte die je kon alloceren voor één bestand, was dusdanig groot dat er slechts een zeer beperkt aantal bestanden zouden kunnen worden bewaard. Met ongeveer 100 studenten en met per persoon ongeveer 10 kleine bestanden zou de diskruimte inefficiënt benut worden en zou een passende hoeveelheid diskruimte (te) duur zijn.

> De hoeveelheid diskruimte was, volgens op onderdelen vage herinneringen, 150 ‘Record Blocks’ (RB’s). Één RB kwam overeen met 56 of 48 ‘Physical Record Units’ (PRU’s). Één PRU kwam overeen met 64 woorden van elk 60 bits. De bits tellend kwam de toegewezen diskruimte, uitgaande van 48 PRU’s per RB, overeen met 150 * 48 * 64 * 60 / 8 ≈ 3,3 [MiB]. Echter, CDC gebruikte een tekenset van slechts 63 of 64 tekens, dus er waren maar 6 bits per teken nodig. In één computerwoord pasten dus 10 tekens. De tekens tellend en één teken gelijkstellend aan één byte (in casu, een vertaling naar ASCII) kwam de toegewezen diskruimte overeen met 150 * 48 * 64 * 10 ≈ 4,4 [MiB].

De meeste bestanden van de studenten waren echter klein, veel kleiner dan één allocatie-eenheid. Jan Kuperus heeft toen gevraagd om dan één groot bestand aan te maken en daarin compact de kleine bestandjes van de studenten op te slaan. Er is dan een programma nodig om die bestandjes-in-een-bestand te beheren. Van het programma om de “bibliotheek” van bestandjes te beheren leven meerdere namen in de herinneringen, zoals ‘pfut’, ‘pfutils’, ‘multi management file system’ en ‘BB’.

De programma’s om de meetgegevens van studenten te bewerken, geschreven in Fortran, werden niet samen met de kleine bestandjes van studenten opgeslagen. Mogelijk werden deze programma's wel op een vergelijkbare manier opgeslagen. De Cyber ‘loader’ werd aangeroepen om een programma en de benodigde brokken uit systeembibliotheken in het werkgeheugen te laden. Een handig en opmerkelijk verschijnsel waren de zogeheten ‘weak references’: deze hoefde bij het starten van het programma nog niet ingevuld te zijn, maar dat kon later alsnog worden gedaan met een aanroep van dezelfde ‘loader’. Dat was niet echt moeilijk. Mogelijk is deze optie gebruikt om werkgeheugen uit te sparen, want ook dat was maar beperkt beschikbaar. Dus pas als je ging plotten werd de daarvoor benodigde programmatuur geladen. Dit programma is heel bescheiden LOS gaan heten, een acroniem voor ‘Library Operating System’.

In het OS van de Cyber 73, ‘SCOPE’ en later ‘NOS/BE’ geheten, was er een onderscheid tussen permanente bestanden en lokale bestanden. Een lokaal bestand bestond alleen tijdens een ‘job’ of tijdens een interactieve sessie op de Cyber 73. Zo gauw de ‘job’ of de sessie ten einde waren, werden alle lokale bestanden die daar bij behoren direct verwijderd. Op de grootte van lokale bestanden waren minder stringente limieten geplaatst. Veel gebruikte opdrachten aan programma BB waren dan ook de kopieeropdrachten tussen lokale bestanden enerzijds en het bibliotheek-bestand anderzijds. BB lijkt functioneel op de ‘archive utilities’, zoals ‘zip’, maar dan zonder compressie.

> Control Data Corporation heeft twee OS-en gemaakt voor de Cyber computers, ‘NOS/BE’ en ‘NOS’. Beide OS-en kenden ‘permanent files’ met vergelijkbare karakteristieken: ze werden toegankelijk na een ‘attach’-opdracht, ze bleven bestaan aan het einde van een sessie en de allocatie-eenheid was vrij groot. ‘NOS’ kende daarnaast ook ‘indirect permanent files’, waarin het gebruikersbestand werd opgeslagen in een groot bestand, zodat de diskruimte beter benut werd. Deze werden toegankelijk door een ‘get’-opdracht. Functioneel leek dat veel op wat programma BB realiseerde.

BB hield dus een administratie bij van welke bestandjes van wie waar stonden met de bijbehorende levensduur. Het hield tevens bij hoe vaak bestandjes gebruikt werden. Op deze wijze konden in onbruik geraakte bestanden gevonden en verwijderd worden. Ook bestanden van stafleden werden op deze wijze opgeslagen. Deze bestanden kregen een langere levensduur.

> Het ACCU was niet zo blij met BB. Tom Hoogenboom had een agressieve vorm van ‘garbage collection’ geïmplementeerd. Als na een (kleine) verandering van een bestand er een stukje in het bestand-met-bestanden ongebruikt was, dan werd het gehele bestand-met-bestanden opnieuw geschreven om alle ongebruikte diskruimte (lege bytes) kwijt te raken. Dit kostte heel veel I/O capaciteit en ‘PP cycles’. *(Had dit te maken met het in rekening brengen van het feitelijke diskgebruik?)*
>
> De standaard levensduur van een bestand van een staflid was initieel 1024 dagen. Jan Kuperus vroeg een keer wat de maximaal mogelijke levensduur is. Gezien het feit dat dat in een register van 18 bits werd opgeslagen, was dat snel uitgerekend. Toen is de levensduur van de bestanden van Jan Kuperus op deze groots mogelijke waarde gezet. Echter, er was geen rekening gehouden met ‘sign bit extension’, waardoor de levensduur negatief werd. In de week na deze verandering verdwenen de bestanden van Jan Kuperus. De historie vertelt niet hoe goed de back-ups waren.

2.1. LIC

Voor de verwerking van de resultaten uit proeven waren na verloop van tijd standaard programma’s, veelal geschreven door studenten, beschikbaar. Dit waren veelal FORTRAN programma’s. De vertaalde versies, ‘LGO’ (voor ‘load-and -go’) in CDC termen, stonden ook in de LOS bibliotheek en kon via LOS gestart worden. Als een LGO geladen werd, maakte LOS een ‘loader request table’ aan, vroeg de ‘loader’ om de LGO en de daarvoor benodigde systeem bibliotheken in het werkgeheugen te laden, waarna LOS weer de controle kreeg. Het kleinst mogelijke deel van LOS bleef in het werkgeheugen staan. De benodigde brokjes van LOS die direct na het laden nodig waren, bijvoorbeeld voor het bijwerken van de statistieken, werden achter het zojuist geladen programma in het werkgeheugen geplaatst. Dus de plek waar het in het geheugen terecht zou komen was a priori onbekend.

Het zou te veel tijd, en mogelijk ook moeite, kosten om de ‘loader’ wederom aan te roepen om een stuk van LOS in het werkgeheugen te laden en de adressen aan te passen aan de locatie waar het terecht gekomen was. Om die reden was *(een groot deel van?)* LOS geschreven als ‘location independent code’ (LIC). LOS was geschreven in machinetaal, en met behulp van de micro en macro faciliteiten in de CDC assembler, “COMPASS” geheten, was het relatief eenvoudig om de programmatuur zo te schrijven dat er geen enkel absoluut adres voorkwam in de instructies, alleen relatieve adressen waarbij in een (B-)register het basisadres opgeslagen was.

> De eerste variant van deze vorm van machinetaal noemden we ‘position independent code’, maar deze naam is niet gehandhaafd omdat de acroniem wat lullig was.
>
> De micro faciliteit in COMPASS hield in dat stukken tekst, mogelijk meerdere regels, op de ene plek in het machinetaalprogramma konden worden aangemaakt en op een andere plek, als ware het gewoon broncode, konden worden ingevoegd. COMPASS was een ‘two pass assembler’, maar met behulp van micro’s is daar effectief een ‘three pass assembler’ van gemaakt. Een derde gang was nodig om een compacte lijst van te laden programma’s te maken, met zowel voorwaartse als achterwaartse verwijzingen.

Het laden van een brokje LOS, geschreven als LIC, bestond uit het aanroepen van het programma om iets van disk te lezen (CIO) waarbij als buffer een locatie direct na het zojuist geladen programma werd opgegeven. Na het vullen van het register met het basisadres kon het worden aangeroepen. Deze manier van het werken bleek razend snel.

> In de initiële versies van LOS waren de foutmeldingen allen in het Engels gesteld. Jan Kuperus heeft een keer gevraagd of de foutmeldingen in het Nederlands gesteld konden worden. Dat is gedaan en ik herinner me vooral nog de foutmelding ‘lader aanvraag tabel vloeit over’.

2.2. Software ontwikkeling

Als ik me het goed herinner is een initiële versie van LOS geschreven in een zomervakantie. Toen eenmaal het ontwerp een beetje duidelijk was ontstond er een werkwijze waarin de ontwikkeling vrij snel ging. In die werkwijze werd letterlijk dag en nacht gewerkt.

Des avonds arriveerde Tom Hoogenboom op Transistorium I, alwaar een stapel listings van de momentane versie van LOS klaar lagen, alsmede een kort verslag van de ontwikkelingen en de problemen. Na een nachtje werken vertrok hij veelal tegen 05:00, maar niet nadat hij een listing had gemaakt van elk van de veranderende brokken en een verslag van de ontwikkelingen die nacht. Rond 08:00 verscheen Wim Nelis op Transistorium I, en kon hij verder gaan waar Tom was gebleven. Tegen 17:00 was Wim aan de beurt om nieuwe listings en een nieuw verslagje te maken.

2.3. Gebruik

Het eigen gebouwde ‘zip’-systeem werd ook het ‘multi management file system’ genoemd en werd aangeroepen met de ‘MM’-opdracht. Echter, de ‘M’-opdracht stuurde een bericht naar de operator van de Cyber 73. Studenten maakten nogal eens de fout om een “M” te weinig in te tikken, dus kwamen ze onverwacht in gesprek met de operator. Er is toen besloten om de naam te veranderen in “BB” (*een acroniem voor BestandsBeheer?*).

2.4. Onderhoud

Met het vorderen van de studies is het beheer van de programmatuur overgedragen aan volgende generaties studenten. We hadden de gewoonte om aan het begin van elk programma een blok commentaar op te nemen, met daarin een beschrijving van de functie, de namen van de auteurs en een beknopt overzicht van de versies en veranderingen. In een latere versie stond in het commentaarblok als verandering vermeld: “This program has been expurgated from the most glaring sins against the English language.”

3. Terminalruimte

De terminalruimte besloeg twee kamertjes aan een uiteinde van het (lange) Transistorium I, op de eerste verdieping, tegenover de lokalen waarin de proefjes werden uitgevoerd. Vanuit de brede hal, die over de hele lengte van het gebouw liep, kon je via een trap naar de terminalruimte. De terminal ruimte bevatte ook nog een deur naar de (smalle) gang op de eerste verdieping.

3.0. Plotter

Naast de 110-baud Teletype rammelkasten hadden we ook één matrix terminal, op wel 300 baud. Met een cassette-unit, waarop je lokaal data kon opslaan of inlezen. Voor precies die snelle terminal

had Jan Kuperus een HP plotter gekocht, met als doel dat studenten een grafiekje met metingen en een trendanalyse (d.w.z. een rechte lijn berekent met de lineaire kleinste kwadraten (LKK) methode) eenvoudig ter plekke konden maken voor hun verslag. Daardoor hoefden ze niet naar het ACCU-gebouw te gaan, bijna 1 [km] verwijderd van Transistorium I, waar de Calcomp plotters stonden. Het eerste gebruik was teleurstellend. De HP-plotter was een vector plotter, maar die Calcomp plotters waren stappen plotters. Het maken van een grafiek op de HP plotter in stapjes van een Calcomp plotter over zo'n langzame verbinding duurde erg (te) lang.

Met een slim algoritme om (veel) stappen naar (minder) vectoren om te zetten en een hack om dat algoritme 'semi-standaard' actief te maken (in de bestaande systeembibliotheken) hebben we een grote glimlach bij Jan Kuperus teweeg gebracht en vele malen een student het loopje naar het ACCU bespaard.

3.1. Reparatie

Een Teletype terminal bestaat voor een groot deel uit elektrische motoren en mechanica, te weten stangen, assen, tandwielen en wat dies meer zij. Op een avond werkte een Teletype terminal niet (*goed?*) meer. In de kast in een hoekje van de ruimte lag ook een 'technical manual' van de terminal, waarin 'exploded view'-tekeningen van het apparaat. Het moest dus mogelijk zijn zelf het apparaat te repareren.

Op die avond kwam een staflid *(Jan Kuperus?)* de ruimte binnen. Hij trof een vloer vol met onderdelen aan, waartussen een paar studenten zaten, die gebogen over de documentatie aan het uitvogelen waren hoe het apparaat weer in elkaar te zetten.

Ik herinner me niet of we er toen in geslaagd zijn de bron van het probleem te vinden, het probleem te verhelpen en een werkende terminal achter te laten.

3.2. Dode toetsen

De Teletype terminals waren robuuste apparaten. Maar ook deze apparaten moesten het soms afleggen tegen studenten. Een defect exemplaar werd door het beherende bedrijf meegenomen ter reparatie en ze lieten een vervangende terminal achter. In die terminal waren magneetjes onder de toetsen gemonteerd. Ze hadden de vorm van een hoekige hoofdletter C. Als een toets werd ingedrukt dan ging een pootje van de magneet langs een reed relais, waardoor de toetsaanslag gedetecteerd werd. Als de toets wat harder werd aangeslagen, dan kon het magneetje breken met als gevolg dat de betreffende toets niet meer werkte. Het ontstaan van dit probleem was goed te merken, want je hoorde een stukje magneet naar beneden vallen. Vooral de magneet onder de 'Enter'-toets had het zwaar te verduren.

De oplossing was om een magneet weg te halen vanonder een toets die niet tot weinig werd gebruikt. Op de dag dat de gerepareerde terminal weer geïnstalleerd werd, hadden we alle weinig gebruikte toetsen al ontdaan van hun magneet. We waren bezig met de vraag welke letter of cijfer nu echt opgeofferd moest gaan worden.

3.3. Papierbandoproller

De Teletype terminals waren voorzien van zowel een papierbandponser als een papierbandlezer. Papierband kon dus gebruikt worden als lokaal stuk geheugen. Papierband werd opgerold bewaard in ronde dozen. Op de terminalkamer stond dan ook een apparaat om een papierband op te rollen. Dit apparaat was solide uitgevoerd en geheel gemaakt van metaal. Desondanks was het apparaat

niet ‘student proof’: op gegeven dag bleek de zwengel verbogen te zijn, zodat deze niet meer rond gedraaid kon worden.

4. Verrekening

Een betalende klus was het maken van een programma om onderwijsinspanningen tussen faculteiten te inventariseren. Dit gaat over colleges gegeven door faculteit A, waarbij een deel van de deelnemende studenten van faculteit B afkomstig is. Die inventarisatie was nodig om te komen tot een onderlinge verrekening tussen faculteiten.

> Een student biologie die een college natuurkunde over meten had gevolgd ventileerde zijn ongenoegen over de gedoceerde werkwijze: “Als je iets meet moet je drie dingen opgeven: de gemeten waarde (duidelijk), de meetfout (we maken geen fouten) en de eenheid (die is voor de hand liggend). Alleen de eerste is dus nodig.”

In ieder geval hebben Cees Visser *(?)* (een scheikundestudent, die vaak in de terminalkamer van het natuurkundepracticum in Transistorium I te vinden was) en Wim Nelis aan deze klus gewerkt, maar mogelijk nog meer studenten.

Ook hier speelde de geringe opslagcapaciteit een rol. Alle gegevens over een brokje werk werden zo compact mogelijk in 60-bit woorden verpakt. Er waren twee, in machinetaal geschreven routines, om dit te bewerkstelligen. Subroutine ‘prop’ schreef de afzonderlijke (Fortran) variabelen in een compact blok, dat naar disk kon worden geschreven. Subroutine ‘rafel’ voerde de omgekeerde bewerking uit. Waarschijnlijk liep de communicatie via een Fortran structuur die ‘common block’ heette.

Er was dus een tabel, noem deze tabel A, die beschrijft wat waar in een compact blok komt te staan of staat. Gaande het project veranderde dat zo vaak, dat er een nieuwe tabel, tabel B, is gemaakt waarin de structuur van de voornoemde tabel A beschreven staat. De bedoeling was om tabel A dan niet meer op te nemen in het programma, maar deze run-time te definiëren. In hoeverre dat echt in productie is gekomen kan ik me niet meer herinneren.

De programmatuur was verder geschreven in Fortran. Cees was erg goed in het bedenken van afkortingen. Ik herinner me nog een subroutine genaamd ‘bagprip’, een acroniem voor “bepaal afgeleide grootheden en print profielen”.

5. Limieten

Als natuurkundestudent dien je zaken te onderzoeken. Dit onderzoek begint natuurlijk met een (nieuwsgierigheids)vraag.

SCOPE, het OS van de centrale computer, stelde een limiet op het aantal lokale bestanden per interactieve sessie. Tom Hoogenboom had ontdekt dat de controle op die limiet pas na het beëindigen van een programma werd uitgevoerd. Hetgeen de vraag opwierp wat de limiet is tijdens het uitvoeren van een programma.

Hij heeft vervolgens een programmaatje geschreven dat een serie *(lege?)* lokale bestanden aanmaakte met de namen AAAAAA, AAAAAB, …, AAAAAZ, AAAABA, AAAABB, enzovoorts. Het bleek dat SCOPE één tabel had waarin de namen van de lokale bestanden van alle gebruikers in vermeld stonden. Op dat moment stopten dus ook alle andere programma’s op de

computer als ze een nieuw lokaal bestand wilden aanmaken. Het programma van Tom is door de operator in het ACCU afgebroken, waarna veel bestanden uit de voornoemde tabel verwijderd werden en de overige gebruikers weer verder konden werken.

Tom heeft daarna het programma opnieuw gestart, voorafgegaan door twee extra opdrachten. Het betrof ten eerste de opdracht ‘DUMP ON’, wat betekent dat het OS een geheugendump van een programma moet maken naar bestand OUTPUT mocht het programma afgebroken worden, en ten tweede de opdracht waardoor het lokale bestand genaamd OUTPUT verwijderd werd. Ook nu liep de tabel vol, opnieuw gaf de operator de opdracht om het programma van Tom af te breken. Maar in dit geval wil het OS lokaal bestand OUTPUT aanmaken, hetgeen niet kan daar de tabel vol is. Het systeem zat dus muurvast. De enige manier om er uit te komen was de computer uit en weer aan te zetten.

Tom is vervolgens naar het ACCU gegaan met het excuus dat hij een tikfoutje had gemaakt in een DO-LOOP waarin bestanden werden aangemaakt. De historie vermeldt niet of het ACCU hem geloofde. Evenmin is duidelijk of de onderzoeksvraag beantwoord is.

6. PDP 8

Het ACCU-gebouw was via een kort gangetje gekoppeld aan het zogeheten mathematisch centrum, waar ook de werkplekken van (een deel van) de ACCU-medewerkers te vinden was. In dit gebouw stond in een hoekje / nisje op de tweede of derde verdieping een niet meer gebruikte minicomputer, een PDP 8. Bij deze computer was papierband een belangrijk (invoer)medium. Het kon (ook) geprogrammeerd worden in BASIC. Tom Hoogenboom heeft een uiterst simpel sorteeralgoritme, de zogeheten ‘bubble sort’, in BASIC geschreven. Behalve simpel is dit algoritme ook traag. Na de invoer van de lijst getallen 50, 49, 48, …, 2, 1 had de PDP 8 ongeveer een kwartier nodig om de volgorde om te keren.

7. Optimalisatie PP benutting

Een herinnering van Edgar Hildering.

Als eerder gemeld, bestond de Cyber 73 computer uit twee ‘Central Processing Units’ (CP’s) en meerdere, zeker 8, ‘Peripheral Processing Units’ (PP’s). Een beperkt aantal PP’s waren continue in gebruik voor een taak (MTR, DSD en 1ND), maar de rest van de PP’s werden pas ingezet naar aanleiding van een verzoek vanuit een CP om een bepaalde taak, bijvoorbeeld I/O, uit te voeren. Het voor die taak benodigde programma werd dan geladen in een PP, gestart en na afronding meldde de PP dat het weer beschikbaar was voor een taak.

Bijgevolg waren er vaak een paar PP’s die niets te doen hadden, ze waren een redelijk deel van de tijd ‘idle’. Dat besef, gekoppeld aan het feit dat de benuttingsgraad van de CP’s erg hoog was, wierp de vraag op of deze ‘idle’ tijd ingezet kon worden voor het doorrekenen van onze practicum resultaten? Tom had inmiddels een manier gevonden om een programma in zo'n PP te laden. Dus aan de slag. Het viel wel op dat het gedrag van de printers een beetje veranderde: de output kwam vertraagd en soms helemaal niet, onze resultaten wel. Dat bleef natuurlijk niet lang onopgemerkt. Tom en Edgar moesten voor een soort 'krijgsraad' verschijnen van heren, die ons nadrukkelijk vroegen 'waar wij in hemelsnaam mee bezig waren!'. Dat was wel duidelijk. Tom verklaarde het door te zeggen dan 'idle' in computers eigenlijk weggegooid geld was. En dat ze blij konden zijn dat wij niet de CPU maar juist de idle tijd van de PP’s aan het gebruiken waren. Toegegeven (ik hoor het hem nog zeggen) er moest nog wel wat geoptimaliseerd worden want printjobs hadden

natuurlijk voorrang, dat stond vast. Ik kan me die rood aanlopende gezichten nog wel herinneren. En eerlijk gezegd had ik het ook best benauwd.

In het hele gezelschap was er een technicus van CDC die eigenlijk helemaal niet zo gestrest was. Nadat we de preek hadden ontvangen, kwam hij op ons toe en vroeg hoe we dat allemaal hadden gedaan en of hij naar onze code mocht kijken. Hij was heel geïnteresseerd. Hij begon te lachen toen we een kwinkslag maakten op het draaiende besturingssysteem: 'we had a NOSe for it!'

> Noot van Wim Nelis: Ik kan me dit voorval niet herinneren. Het impliceert het omzeilen van wat beveiligingen, want een PP-programma werd geladen vanuit een systeembibliotheek, die voor gewone gebruikers niet benaderbaar was. Maar het wachtwoord UNIVPERM zou gebruikt kunnen zijn. Een PP-programma is notoir moeilijk om te testen. Als er iets mis ging in een PP-programma kon je een (octale) geheugendump op een printer laten maken tijdens het herstarten van de computer. Een PP heeft slechts 4096 woorden werkgeheugen, waarvan een deel ook nog continue werd bezet door een brok basisprogrammatuur, ‘PP resident’ geheten. Een PP kon alleen in machinetaal geprogrammeerd worden. Als ik me het goed herinner waren er geen ‘floating point’ instructies: zelfs het manipuleren van een 18-bit integer (adres in centraal geheugen) met alleen 12-bit registers was niet beschikbaar. Al met al een niet-triviale klus.

8. Vervolg gebruik Cyber 73-28

Wim Nelis is na het afronden van zijn studie in 1979 gaan werken bij het Nationaal Lucht- en Ruimtevaartlaboratorium (NLR). Daar stond ook een computer CDC, een Cyber 72. In 1980 is deze vervangen door een Cyber 73-28, dezelfde computer die tijdens de studie van Wim aan de RUU bij het ACCU stond.

Bij de Cyber bestond het (koud) starten van de computer uit het lezen van het ‘deadstart panel’, een matrix van 16 *(?)* * 12 schakelaars, deze te laden in het werkgeheugen van PP0 en vervolgens PP0 te starten. Elke schakelaar kwam overeen met één bit. Elk rijtje van 12 schakelaars kwam in één woord (van 12 bits) terecht. Het ‘deadstart panel’ bevatte dus een minuscuul programmaatje dat in staat was om een paar kaarten van de kaartlezer te lezen (o.i.d.), waarmee een iets uitgebreider programmaatje werd geladen, die wat meer kon, bijvoorbeeld een tape lezen waarop het besturingssysteem stond.

In het ‘deadstart panel’ staat dus op een paar plekken het adres van het apparaat waar het het volgende programmaatje van leest. Bij de ingebruikname van de Cyber 73 op het NLR werkte het starten niet goed: er werd niets gelezen van de kaartlezer. Na de nodige controles, waaruit bleek dat de schakelaars echt goed stonden, kreeg iemand een helder idee. In de 7 jaar sinds de Cyber 73 bij het ACCU in gebruik was genomen, zijn de schakelaars nimmer geschakeld tot het bij het NLR stond. De contacten van de schakelaars waren waarschijnlijk gecorrodeerd. Alle schakelaars zijn toen een tiental malen geschakeld, om de contactpunten te ontdoen van roest. Daarna is het initiële programmaatje weer ingesteld op het ‘deadstart panel’ en startte de computer als verwacht.