stichting
mathematisch
centrum

MC

AFDELING INFORMATICA IN 12/76 OKTOBER

D. GRUNE

ERVARINGEN MET DE OVERDRACHT VAN DE ALEPH-COMPILER VAN SCOPE 3.4.1 NAAR SCOPE 3.4.4 OFTE WEL DE SOFTWARE-CRISIS WOEDT

2e boerhaavestraat 49 amsterdam

Printed at the Mathematical Centre, 49, 2e Boerhaavestraat, Amsterdam.

The Mathematical Centre, founded the 11-th of February 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications. It is sponsored by the Netherlands Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O), by the Municipality of Amsterdam, by the University of Amsterdam, by the Free University at Amsterdam, and by industries.

AMS(MOS) subject classification scheme (1970): 68A15

ACM-Computing Reviews-categories: 4.43, 4.6

Ervaringen met de overdracht van de ALEPH-compiler van SCOPE 3.4.1 naar SCOPE 3.4.4
ofte wel
De software-crisis woedt

door

D. Grune

SAMENVATTING

De moeilijkheden die zich voorgedaan hebben bij een eenvoudige programma-overdracht worden uitgebreid beschreven, geanalyseerd en in klassen ingedeeld. Voor deze klassen van problemen worden oplossingen gesuggereerd.

TREFWOORDEN: *overdraagbaarheid, ALEPH, SCOPE operating system.*

1. INLEIDING

Het overdragen van de ALEPH-compiler van het operating system SCOPE 3.4.1 naar SCOPE 3.4.4 is niet van een leien dakje gegaan, en het is interessant te bekijken waarom niet. Het is beslist onrustbarend dat het overdragen van een als zeer overdraagbaar ontworpen compiler van de ene versie van een operating system naar een "slechts weinig verschillende" latere versie al één man-maand kost.

Nu hoeven niet alle specifieke bokken die hierbij geschoten zijn voor het nageslacht bewaard te worden, maar de hele operatie heeft een paar interessante aspecten die tot algemenere gedachten, of misschien zelfs conclusies, leiden.

Bij het doorlezen van de SARA-publicatie [1] die de gebruikers van de op til zijnde verandering op de hoogte bracht, kwamen als voor ALEPH belangrijk de volgende drie onderwerpen naar voren:

1e. de parameters van RECOVR zijn veranderd
2e. MEMORY werkt anders
3e. het formaat van Z-type records is veranderd.

Uit een toevallig gesprek met iemand van SARA bleek dat er ook bij de aanroep van de COMPASS-assembler moeilijkheden te verwachten waren.

Bij de overdracht van de ALEPH-compiler werd bijna alle tijd en moeite besteed aan de volgende drie onderwerpen:

1e. het formaat van Z-type records
2e. de aanroep van COMPASS
3e. onverklaarde fouten.

Dus slechts één van de drie werkelijke problemen was voorspeld en slechts één van de drie voorspelde problemen deed zich voor.

Ik zal nu de vijf genoemde problemen één voor één de revue laten passeren.

2. RECOVR

De SARA-publicatie [1] deelt mede dat enkele error codes van RPV (reprieve-function) veranderd zijn. Zorgvuldige vergelijking van oude en nieuwe documentatie levert geen verschillen op. De door ALEPH gebruikte aanroep van RECOVR blijkt goed te functioneren.

Resultaat: een blijvend gevoel van wantrouwen jegens het ding: misschien doet het toch niet meer altijd precies wat we denken (maar welke systeem-functie doet dat wel?).

3. MEMORY

De functie MEMORY dient om de geheugenruimte uit te breiden of in te krimpen tot de gevraagde waarde, als dat kan; als het niet kan gebeurt er niets.

In het oude systeem was dat de enige mogelijkheid; als het niet ging, probeerde het ALEPH-systeem het met minder en minder, net zolang tot het wel ging.

In het nieuwe systeem kan via een speciale optie gevraagd worden hoeveel er maximaal gevraagd kan worden; dit maakt het proberen van telkens kleinere eisen overbodig.

We hebben besloten van deze nieuwe optie geen gebruik te maken: de winst is gering, het is werk en het maakt het onmogelijk de code ooit nog op SCOPE 3.4.1 te draaien.

4. Z-TYPE RECORDS, HET EERSTE ECHTE PROBLEEM

4.1. In het SCOPE-systeem zijn de karakters (6 bits elk) in een file (die

uit machine-woorden van elk 60 bits bestaat) met z'n tienen in een woord gepakt. Elke regel bestaat een geheel aantal woorden (zodat een FORTRAN READ niet midden in een woord blijft steken).

Bij deze opzet rijzen direct twee vragen: hoe vind ik dat laatste woord en wat gebeurt er met regels die niet genoeg karakters hebben om precies een geheel aantal woorden te vullen.

Het laatste woord is te herkennen aan het feit dat de twee meest rechtse karakters allebei de representatie 00 octaal hebben. Echter, een dubbele punt heeft ook de representatie 00 octaal. Dit heeft tot gevolg dat een kaart die twee dubbele punten heeft in bijvoorbeeld kolom 9 en 10, als twee regels opgevat wordt en dat de dubbele punten tussen de wal en het schip raken. Om deze misère te vermijden heeft CDC een sysgen-optie geschapen waarbij het procent-teken wordt opgedoekt, de dubbele punt naar de plaats van het procent-teken verhuist en de representatie 00 octaal zonder karakter blijft (en dus niet ingelezen kan worden); het vóórkomen van 00 octaal ergens anders in een woord (of als meest rechtse karakter wanneer het één na rechtse niet nul is) blijft legaal. SARA heeft deze optie gekozen.

Uit het bovenstaande volgt dat regels altijd 10n + 8 karakters lang zijn. Aangande de Procrustes-methoden toe te passen op regels die niet aan de eisen voldoen, verschillen het oude en nieuwe systeem van mening. In het oude systeem werden ze aangevuld met spaties (55 octaal), in het nieuwe met nullen (00 octaal), zoals beschreven in de SARA-handleiding [1]. Dit is onder andere gedaan omdat in het oude systeem bij interactief gebruik de lijnen onnodig belast werden met al die spaties.

Het karakters-gewijs inlezen van een file in het nieuwe formaat geeft aanleiding tot een onverwacht probleem. Stel een regel is 10n + 9 karakters lang. Dan wordt hij gerepresenteerd door n woorden met elk 10 karakters, 1 woord met 9 karakters, en helemaal rechts 00 octaal en 1 woord met allemaal octale nullen. Zijn we nu teken voor teken het woord met de 9 karakters en een 00 aan het lezen en zijn we bij het tiende karakter gekomen, dan zijn er voor de 00 twee mogelijkheden: óf het is het eerste vulsel (en dan is het volgende woord 0), óf het is een (in deze positie legaal) karakter 00 octaal (en dan is het volgende woord niet nul). Dit is allemaal wel op te lossen, maar wordt nog wel gecompliceerd door het feit dat het ALEPH run-time systeem het zo nu en dan nodig vindt het laatst gelezen teken terug te drukken, en dat kan dan b.v. bovengenoemde patiënt zijn. Het toont

wel aan dat er aan de invoer-routines fors gesleuteld moet worden.

Bovenstaande verwikkelingen waren voor Wirth [2] aanleiding zijn mening over file-systemen te herzien en de definitie van PASCAL te wijzigen; dus we zijn niet alleen.

We hadden ons dit probleem kunnen besparen door van meet af aan de Record Manager [7] te gebruiken. Aan het gebruik van de Record Manager door iets anders dan FORTRAN en COBOL kleven echter zoveel bezwaren dat onze huidige problemen daarbij verbleken. De Record Manager gaat er namelijk van uit dat hij voor een FORTRAN- of COBOL-programma werkt: hij doet zijn eigen geheugen-allocatie op zijn eigen manier (het programma groeit toch niet) en hij leest hele regels in één keer. Over efficientie praat ik niet. Ter illustratie: onder SCOPE 3.4.4. gebruiken systeem-programma's (b.v. de COMPASS-assembler) die vroeger de Record Manager gebruikten, nu weer CIO (Central Input Output), net als ALEPH.

Het nieuwe formaat werd in twee (man-) dagen geïmplementeerd en getest. Hierbij bleek dat het systeem zich niet geheel en al gedraagt zoals hierboven beschreven; we hebben de volgende afwijkingen waargenomen (waarvan alleen de eerste beschreven blijkt te zijn en wel in het Record Manager Reference Manual [7]):

1e. alle regels die eindigen op een : worden verlengd met één spatie, kennelijk een atavism uit de tijd dat de dubbele punt 00 octaal was.

2e. sommige regels die eindigen op een punt worden met een spatie verlengd. De reden is onduidelijk, de regelmaat ook.

3e. lege kaarten worden meestal ingelezen als twee spaties. Dit is nuttig voor als het kaartendek direct naar de printer moet en al "control karakters" bevat. De eerste spatie komt dan goed van pas; het is niet duidelijk waar de tweede spatie voor dient.

4e. een dood-enkele lege kaart wordt ingelezen als 00 octaal gevolgd door een spatie. Dit heeft ons gedwongen in de inlees-routine "get char" alle 00-karakters te vervangen door spaties, waardoor zowel informatie als efficientie verloren gaat.

Al deze afwijkingen zijn op zichzelf niet erg (afgezien van punt 4) maar doen zich wel voor als fouten in "get char" en "put char" en vereisen

daarom analyse, met de daaraan verbonden tijd-verspilling.

Het probleem is hier: een schijnbaar eenvoudig, maar op de keper beschouwd grillig systeem met ontoereikende documentatie.

4.2. Toen de juiste "get char" en "put char" eenmaal geïmplementeerd waren, bleek dat de ALEPH complier niet meer bereid was ook maar één ALEPH-programma als correct te accepteren. De reden bleek te zijn dat de compiler na elk vet woord nog minstens één karakter wil zien, om zijn buffertje mee te vullen. Dus ook na end.

Nu staan er in het oude systeem na end (gespeld 'end') nog drie spaties, maar in het nieuwe systeem staat er niets meer. Om het weer goed te laten werken, was een wijziging in de compiler nodig.

Dit verschijnsel werpt een bepaald licht op overdraagbaarheid. We hebben hier de situatie dat een fout in de compiler van zodanige aard is dat hij enerzijds op het oude systeem niet redelijkerwijs ontdekt had kunnen worden en anderzijds op het nieuwe systeem functioneren totaal onmogelijk maakt. In de praktijk ontstaat zo een debugging-loop tussen gever and ontvanger van de compiler, een hoogst ongewenste situatie.

Hierbij moet nog aangetekend worden dat toen een van de leden van de ALEPH-groep een jaar geleden de code voor het lezen van vette woorden doorlas, hij opgemerkt heeft dat het hierboven genoemde verschijnsel op zou kunnen treden. Er werd toen besloten er niets aan te doen, omdat:

1e. het oplappen van de code geen fraai resultaat zou leveren,
2e. het volledig corrigeren van de code meer mankracht vroeg dan beschikbaar was,
3e. de klacht niet dringend was, en er urgentere dingen waren.

Problemen van deze aard kunnen alleen voorkomen worden door te zorgen dat de geleverde code geen fouten bevat, en aangezien we allemaal weten dat dat niet kan, zullen we er mee moeten leven. Wel is het duidelijk dat fouten in een voor distributie bestemd systeem veel hinderlijker zijn dan in een systeem voor plaatselijk gebruik, en het loont dan ook zeker de moeite, zo goed mogelijke code te leveren.

In het ALEPH-kader zie ik daarvoor twee mogelijkheden, ook tegelijkertijd toepasbaar:

1e. alle code wordt door minstens twee personen in directe samenwerking geschreven; de huidige situatie, waarbij één persoon de compiler beheert, en één het run-time systeem, is belachelijk. Deze aanpak zal wel duurder zijn, maar het is moeilijk te schatten hoeveel. Enerzijds hebben deze twee personen elk minder werk, anderzijds hebben ze tijd en energie nodig om het over alles eens te worden.

2e. er wordt tegelijkertijd op twee volkomen verschillend systemen geïmplementeerd.

Overigens zijn we niet de enigen met dit probleem: de Software Engineering Group of the Dept. of Electrical Engineering, University of Colorado, Boulder, Colorado, schrijft in [3]:

We have completed the bootstrap of PASCAL J to the Xerox Sigma 3, and in the process uncovered and corrected a number of errors in the compiler.

4.3. Het ALEPH run-time system bevat een routine "get int" die een integer leest. Deze bleek op het nieuwe systeem plotseling het laatste getal van de file niet weer te lezen. Ook hier hetzelfde, als er een spatie achter had gestaan was het goed gegaan. Deze routine was door één van de leden van de ALEPH-groep speciaal ten behoeve van de overgang naar het nieuwe systeem nog eens grondig doorgelezen, en hij heeft het niet gezien. Ook hier geldt: óf alleen genieën in dienst nemen, óf de code door twee personen laten beheren.

4.4. Moraal:

1e. Het vervaardigen van overdraagbare soft-ware is veel duurder dan van locale soft-ware: alle code moet door twee personen beheerd worden en/of op twee machines geïmplementeerd worden.

2e. De huidige situatie is ontoereikend en kan niet tot resultaten leiden.

3e. Er is niets urgenters dan het corrigeren van gevonden fouten: vandaag struikel je erover, maar vannacht in 't donker breek je er je benen over.

5. DE AANROEP VAN COMPASS

5.1. Bij het ontwerp van de CDC-versie van de ALEPH-compiler werd besloten om de code-generatie via COMPASS, de assembleer-taal van de Cyber, te laten verlopen, omdat we hierdoor een goed gedefinieerd en leesbaar interface krijgen. Dan ligt het voor de hand deze COMPASS tekst door de COMPASS assembler te laten verwerken, zodat we ons het bit-gepriegel besparen, nodig om zelf "relocatable binary" te produceren.

Dit laatste was geen goed idee, nog afgezien van de teleurstellende efficientie van de COMPASS assembler: meer dan 30% van de totale tijd om een ALEPH programma te vertalen gaat in de COMPASS-assembler zitten. Dit komt (waarschijnlijk) omdat hij voortdurend test op de aanwezigheid van allerlei uitzonderlijke opdrachten, die door de ALEPH compiler nooit geproduceerd worden.

5.2. Wil de ALEPH compiler, zoals elke nette compiler, "relocatable binary" afleveren, dan moet hij zelf de COMPASS-assembler aanroepen. Nu is COMPASS een "absolute overlay" in NUCLEUS en er staat in het Loader Reference Manual [4] hoe je zo'n ding moet laden en starten. Pogingen hiertoe faalden. En bij SARA wist men van niks.

Bij een, verder toevallig, bezoek aan CDC in Rijswijk hoorde ik wat er niet in het Loader Reference Manual staat, n.l. dat de Loader, als hij een control-card load uitvoert, de registers A0 en X0 op de groottes van resp., Central Memory en Extended Core Storage zet, maar dit niet doet wanneer hij een user-initiated load uitvoert. En de COMPASS assembler rekent er wel op. Na de betreffende wijziging in het aanroepende programma liep de assembler.

Het probleem was hier: slechte documentatie.

Een lopende assembler alleen is niet genoeg: we moeten hem kunnen vertellen waar de invoer staat, waar de uitvoer heen moet, enz. We moeten parameters mee kunnen geven.

In het SCOPE Reference Manual [5] staat waar de parameters bij de aanroep van een programma staan, namelijk in locaties 2B tot 53B, in gecodeerde vorm; en de codering is ook beschreven. Parameters verschaft in deze vorm drongen echter niet tot de assembler door.

Na in een octale dump (op dit niveau niet "het laatste redmiddel" maar het normale hulpmiddel) geconstateerd te hebben dat de parameters er toch echt stonden, begreep één onzer in een helder ogenblik wat er gebeurde: de assembler analyseerde het elders in het geheugen aanwezige control-card

image opnieuw en bouwde zo zijn eigen parameters op!
Het probleem werd opgelost door het toevoegen van een forse lap code om een namaak control-card image te construeren.

Het probleem was hier: een stuk systeem soft-ware maakt geen gebruik van het standaard interface maar knoeit zelf wat.

In bovenbeschreven vorm is de COMPASS assembler ongeveer een jaar door de ALEPH compiler gebruikt. In een later gesprek met iemand van CDC werd mij duidelijk gemaakt dat het zo helemaal niet had mogen kunnen: de COMPASS-overlay heeft een zodanig "access-level" dat hij alleen op een orthodoxe control-card aangeroepen mag worden en niet intern door een user-initiated load. Dus óf het "access-level" is bij SARA verkeerd ingesteld, óf de loader test niet goed (óf zegsman heeft ongelijk). In Utrecht bij het ACCU bleek de aanroep niet te werken.

In dit licht bezien snijden onze klachten geen hout: een programma dat alleen direct door de loader aangeroepen kan worden mag best bijzondere afspraken met de loader hebben en het niet volgen van standaarden op plaatsen waar gebruikers die denken dat die standaarden wél gebruikt worden, tóch niet kunnen komen, is niet verboden (maar wel smerig). Deze filosofie houdt in dat er geen door ALEPH aan te roepen COMPASS is, en dat we ons vergisten, toen we dachten dat we de COMPASS tekst door de COMPASS assembler konden laten vertalen. Het is een opvatting.

5.3. Het is interessant om even te bekijken hoe andere compilers dit problem oplossen.

De FORTRAN-compiler genereert vereenvoudigd COMPASS, samen met een lijst van labels met hun addressen. Daarna wordt, uitgaande van deze informatie (waarvan het COMPASS-deel goed gedocumenteerd is, maar het formaat van de label-lijst helemaal niet) in één scan relocatable binary gemaakt, door een in de FORTRAN compiler ingebouwd assemblertje, FAX. Deze FAX hebben we niet willen gebruiken omdat de documentatie niet duidelijk was en omdat we dan te veel afhankelijk zouden worden van (kleine) wijzigingen in het FORTRAN/FAX interface dat buiten onze greep ligt.

Op verzoek (C-option) is de FORTRAN compiler bereid de geproduceerde COMPASS tekst door de echte COMPASS assembler te laten verwerken. Hiertoe zet hij een "communication area" op, laadt het COMP3$-overlay uit NUCLEUS (dat is het eerste overlay van de COMPASS-assembler) en start hem. Dit is

(waarschijnlijk) de overigens nauwelijks gedocumenteerde, officiële manier om COMPASS aan te roepen. De main overlay van de COMPASS-assembler werkt zelf ook zo.

De PASCAL-compiler en de beide ALGOL 60-compilers genereren direct binaire code.

De ALGOL 68-compiler genereert intern een stroom "ICF-macro's", ongeveer een nette assembleertaal voor een "machine-onafhankelijke Cyber". Deze wordt dan door de twee laatste (gemodificeerde) scans van de SYMPL-compiler geoptimaliseerd vertaald. Op verzoek wordt parallel een COMPASS-achtige listing gemaakt. Wij hebben besloten geen ICF-macro's te gebruiken, gedeeltelijk omdat ICF-code tamelijk moeilijk te genereren is (maar wel zeer goede resultaten geeft!) en gedeeltelijk omdat we dan weer te afhankelijk zouden worden van een zeer speciaal stuk soft-ware.

De PL/I-compiler maakt gebruik van de COMP3$-overlay, net zoals FORTRAN.

5.4. Uit het bovenstaande blijkt wel duidelijk wat de fatsoenlijke manier is om een COMPASS-tekst vertaald te krijgen: gebruik COMP3$. En toen we hoorden (mondeling) dat op het nieuwe systeem onze aanroep van COMPASS niet langer zou lukken vanwege de access-levels en toen we bij proberen op het nieuwe systeem de foutmelding LDV ERR 0003 kregen, dachten we onszelf een gunst te bewijzen door over te gaan op de "officiële" COMP3$.

Bij navraag bij SARA bleek de beste documentatie de (in principe geheime) listing van de COMPASS-assembler te zijn. Daar zit wat in: er is niets wat zó precies beschrijft wat de assembler doet! Zoals verwacht bleek de assembler te bestaan uit een kort programmaatje (2000 tot 3000 kaarten), verder driver genoemd, dat een communication area COMPCOM vult met informatie en dan COMP3$ aanroept (althans in grote lijnen). De driver bevatte inderdaad de gewraakte code voor het opnieuw analyseren van het controlcard image.

COMP3$ gaat er dan van uit dat COMPCOM precies op locaties 101B - 177B staat (De B in deze en verdere adressen betekent dat ze als octale getallen genoteerd staan.). Verder viel het op dat de COMPASS-driver een tekstuele kopie van SYSR (d.w.z. de routines SYS=, RCL=, WNB= en MSG=, voor systeemcommunicatie) bevatte, iets waarvan ik verwacht had dat het er door de (linking) Loader bijgehaald zou worden. Reden: alle CDC soft-ware wordt als "absolute

assembly" geassembleerd en is daarna ongeschikt om door de (linking) Loader nog met iets anders gelinkt te worden. Dus alle benodigde routines moeten tekstueel aanwezig zijn, telkens weer, in elk stuk systeem-software. Dit is hoogst ongelukkig.

Maar, de driver van de ALEPH-compiler (het programmaatje dat de twee scans van de ALEPH-compiler aanroept en daarna COMPASS) is relocatable binary en hoeft zich van deze ongelukkige aanpak dus niets aan te trekken. Ja, dat dacht U.

Het viel op dat COMP3$ niet tekstueel een kopie van SYSR in zich heeft en toch heus wel systeem-communicatie bedrijft, terwijl het bovendien een absolute assembly is. Verklaring: COMP3$ wordt samen met de COMPASS-driver in één absolute assembly vertaald, en maakt gebruik van de SYSR-routines in de driver, gewoon omdat hij weet dat het adres van de SYS=-routine 251B is, enz. En iedereen die COMP3$ wil gebruiken moet dus op 251B het begin van de SYS=-routine hebben. Dus de ALEPH-driver moet tekstueel SYSR in zich hebben. Dit alles staat nergens gedocumenteerd, zelfs niet in de COMPASS-listing.

Eenmaal gewaarschuwd vonden we meer: de File Environment Tables (FETs) van input-, output- en binary file moeten op vaste plaatsen staan.

Het probleem is hier: een hoogst twijfelachtige en bedrieglijke programmeertechniek gecombineerd met een volslagen gebrek aan toegankelijke documentatie. Dit is voor eigen soft-ware draaglijk, voor andermans soft-ware onaanvaardbaar.

5.5. We hebben de tekst van COMPCOM en SYSR door bereidwillige medewerking van iemand van SARA in onze ALEPH-driver over kunnen nemen, en hadden toen nog maar een paar moeilijkheden.

COMP3$ wil weten hoeveel ruimte hij heeft en hoeveel hij mag gebruiken. Deze parameters staan in COMPCOM. Uit de listing bleek dat hij ongeveer 32400B nodig heeft. Dus gaven we hem 34000B met uitbreiding tot 55000B. COMP3$ schrijft dan op geheugenplaats 34000B en die is er niet (aangezien het geheugen bij 0 begint te tellen). Dus gaven we hem 34100B en vertelden hem dat hij 34000B had. Hij leest dan geheugenplaats 44644B. Dus gaven we hem 55000B en vertelden hem dat hij 34000B had. Dan werkt hij. Niet alleen gegeven paarden maar ook duurbetaalde assemblers moet men niet in de bits

kijken.

De gekregen tekst bevatte ook code voor het op de juiste plaats genereren van de FET's (File Environment Tables) voor de drie files die COMP3$ gebruikt: input, output en binary. Deze werkte niet, omdat de buffers voor de output file en de binaire file over elkaar heen gelegd werden. Ik neem aan dat de COMPASS-driver code bevat die dit weer oplapt, en ook dit kan uitgezocht worden, maar er is een grens.

Nog een laatste voorbeeld: de COMPASS-driver laadt COMP3$, kladt er in op een hem bekende plaats en start hem dan. De driver weet dat op locatie zoveel in overlay COMP3$ de naam van de "output file voor noodgevallen" (O-optie van COMPASS) staat. Het probleem is hier: heb een goed gedefinieerd interface (COMPCOM) en gebruik dat op één of twee plaatsen na.

5.6. De totale hoeveelheid moeite besteed aan het verwerken van COMPASS tekst door de ALEPH-compiler schat ik op drie man-maanden van hooggekwalificeerd personeel. Dit is volledig in overeenstemming met de wet van Cheops: alles duurt minstens twee keer zolang als verwacht en kost minstens drie keer zoveel [6].

Er rijzen hierbij een aantal vragen. Ten eerste: hadden we er misschien niet aan moeten beginnen? Met andere woorden, hadden we ons op het standpunt moeten stellen dat de ALEPH compiler alleen maar COMPASS produceert, en het aan de gebruiker moeten overlaten om van dit COMPASS relocatable binary te maken? Elke standaard (en bijna elke niet-standaard) compiler op dit systeem levert relocatable binary, en als wij de kosten daarvan niet willen dragen, betekent dat dat we vinden dat programmeren te duur is om het goed te doen.

Ten tweede: had het langs de weg van een aanroep van de COMPASS-assembler goedkoper gekund? Ik zie niet dat we in het bovenstaande vermijdbare fouten gemaakt hebben. Het project heeft het karakter van experimenteel onderzoek aangenomen, waarbij we over het algemeen, door inzicht en geluk, goed gekozen hebben.

Ten derde: had het langs een andere weg beter gekund? Wanneer we van meet af aan die drie man-maanden besteed hadden aan het schrijven van een zeer eenvoudige assembler in ALEPH, voor een subset van COMPASS, b.v. zoals gedefinieerd voor de FAX-assembler (zie 5.3.), dan hadden we nu een

assembler gehad die (waarschijnlijk) efficienter geweest zou zijn, en die we, wat belangrijker is, in eigen beheer gehad zouden hebben. Nu zitten we met een aanroep van COMPASS die we (a) niet geheel begrijpen (zie het eerste probleem in 5.5.), en die ons b.v. bij grote programma's voor onverwachte problemen zou kunnen stellen, die (b) niet te documenteren is en dientengevolge (c) moeilijk toegankelijk is te maken voor andere leden van de groep, en (d) bij elke systeemwijziging weer op losse schroeven komt te staan.

Al deze klachten zouden komen te vervallen als we zelf een assembler geschreven zouden hebben. Overigens zou ook deze assembler onderhevig zijn geweest aan de wet van Cheops: tweemaal zo lang en driemaal zo duur als verwacht.

Het probleem is hier: we hebben te veel vertrouwd op andermans bestaande soft-ware, in die richting gelokt door de schijnbare, gemakkelijke beschikbaarheid van de COMPASS-assembler.

Wie in een overdraagbaar bedoeld systeem (overdraagbaar van versie A van een Operating System naar versie B van datzelfde OS) soft-ware van iemand anders gebruikt, bouwt zijn huisje op het ijs. Die iemand kan b.v. het interface drastische wijzigen, of zijn soft-ware niet naar versie B overdragen, b.v. omdat hij weg is, of zijn baas belangrijkere dingen voor hem heeft.

5.7. Hoewel de inhoud van de vorige alinea op zich juist is, is het advies dat erin gegeven wordt in uiterste consequentie onbruikbaar: het is onmogelijk te programmeren zonder soft-ware van een ander te gebruiken. Dit werpt de vraag op wat dan wel een bruikbare stelregel zou zijn.

Het bezwaar dat we tegen het gebruik van andermans soft-ware hebben, is dat hij de specificaties en het interface kan wijzigen, en ons daarmee veel werk bezorgt. Dit suggereert twee oplossingen:

a. gebruik alleen faciliteiten waarvan de specificaties weinig zullen veranderen (integer optelling)
b. gebruik de faciliteiten zo dat wanneer de verandering in specificatie komt, de herstlwerkzaamheden beperkt zijn (b.v. een leesroutine voor karakters die eerst wel en later geen spaties aan het eind van een regel levert).

Dit brengt ons tot wat naar mijn mening de kern is van het schrijven van overdraagbare programmatuur:

Soft-ware is meer overdraagbaar naarmate hij minder gevoelig is voor de onvermijdelijke veranderingen in de omringende soft-ware die hij noodzakelijkerwijs gebruiken moet.

Om dit doel na te streven moeten we van elke externe faciliteit die we gebruiken een schatting maken van mogelijke redelijke veranderingen in de specificatie, en daarvoor het programma minder gevoelig (gemakkelijker aanpasbaar) maken. Of, positiever gezegd, een schatting maken welke aspecten niet zullen veranderen, en alleen deze gebruiken.

Een mal voorbeeld: als we er rekening mee houden dat op een dag de karakters in een regel van achteren naar voren ingelezen zullen worden, is het beter om een compiler regelsgewijs te laten lezen dan karaktersgewijs. Een compiler die regels tegelijk inleest is gemakkelijker aan de nieuwe situatie aan te passen dan één die overal losse karakters leest.

Bij de aanroep van de COMPASS-assembler veranderden de specificaties en waren de herstelwerkzaamheden omvangrijk. Volgens bovenstaande criteria was het dus een slecht stuk soft-ware om te gebruiken in een (van SCOPE 3.4.1 naar SCOPE 3.4.4) overdraagbare compiler. Maar dat wisten we pas naderhand.

5.8. Toen we de assembler uitendelijk lopende hadden, bereikte ons van iemand van SARA het bericht dat uit onderzoek bij hun was gebleken dat de vroegere directe aanroep van de COMPASS-assembler ook op het nieuwe systeem bruikbaar is, mits men het "library-bit" (bit 18 van woord 65B) aanzet.

Deze hele gang van zaken doet aan experimenteel biologisch onderzoek denken: het systeem is zo groot en in diepste wezen zo mysterieus dat precieze eigenschappen alleen door onderzoek bepaald kunnen worden. Van enig "boven de materie staan" is geen sprake, we staan er midden in. Zoals bij veel experimenteel onderzoek is de uitendelijke oplossing veel eenvoudiger en ook anders dan verwacht, maar zou (waarschijnlijk) nooit gevonden zijn zonder de lange weg te gaan.

De tijd besteed aan het construeren van de aanroep van COMP3$ is niet geheel verspild. Ten eerste konden we niet van te voren weten dat iemand iets beters zou vinden, ten tweede hebben we er veel van geleerd (zie

hoofdstuk 7) en ten derde laat deze aanroep ons, in tegenstelling tot de directe aanroep van COMPASS, de mogelijkheid, na afloop hulp-files terug te geven, de object-file bij te schaven, enz., allemaal nuttige zaken.

6. MYSTERIES

6.1. In één van de test-programma's die op het nieuwe systeem gedraaid werden, werd een variabele die in het programma van 1 tot 12 loopt, aangetroffen met de waarde 20. De fout was reproduceerbaar, en we konden geen verklaring vinden. Na het corrigeren van de elders in het programma gebruikte "get int" (zie 4.3.) verdween het verschijnsel (post sed non propter?). Er is geen enkele zichtbare relatie tussen deze variabele en get int.

In een poging de fout te analyseren hebben we toen een gesimuleerde foutieve "get int" gebruikt, maar nu traden alleen de verschijnselen van de foute "get int" op en de variabele kreeg de juiste waarde. Uit gebrek aan mankracht hebben we besloten dit niet verder uit te zoeken.

6.2. Tijdens het testen op het nieuwe systeem kwam de ALEPH-compiler met de foutmelding dat er in de tweede scan ergens een pointer niet deugde. Omdat hij bijna wél deugde, hebben we het probleem vrij grondig geanalyseerd en niets gevonden. Na het weekend was het verschijnsel verdwenen. Geen commentaar.

6.3. Deze geest-verschijningen zijn buitengewoon hinderlijk en hebben ons relatief veel tijd gekost die in wezen verspild is. Oorzaken zouden kunnen zijn:

1e. hard-ware fouten en systeem fouten
2e. subtiele misverstanden onzerzijds over wat systeem routines doen (al dan niet gedocumenteerd)
3e. subtiele fouten in de logica van onze programma's

of andere oorzaken.

Hoe het ook zij, we moeten er op rekenen dat de ontvanger van portable

soft-ware ook met deze problemen te maken zal krijgen. Het enige wat we voor hem kunnen doen is hem soft-ware geven waarin hij zoveel vertrouwen krijgt dat dit soort verschijnselen hem niet al te zeer verontrusten.

7. SAMENVATTING

7.1. We zijn de laatste maand bezig geweest met een minuscuul portabiliteits-projectje, waaruit zich de volgende conclusies opgedrongen hebben.

I. Overdraagbare soft-ware is veel duurder dan locale want:

a. de correctheids-eisen zijn veel zwaarder: een onmerkbaar foutje op het ene systeem kan draaien op het andere systeem bijna onmogeljik maken (zie 4.2., 4.3., 6.3.).

b. de documentatie moet veel beter zijn.

II. Overdraagbare soft-ware moet zeer kieskeurig zijn in het gebruik van soft-ware van anderen, en bovendien weinig gevoelig zijn voor veranderingen daarin (zie 5.6., 5.7.).

III. Overdragen van soft-ware naar een slecht gedocumenteerd operating systeem is een bijna onmogelijke taak (zie 4.1., 5.2., 5.5.).

7.2. Voor de zo gestelde problemen zouden de volgende oplossingen gebruikt kunnen worden.

Ad Ia, de correctheid:

Alle code: compiler, driver en run-time systeem, moet door minstens twee personen beheerd en grondig begrepen worden.
Het systeem moet op minstens twee machines geïmplementeerd worden.
Het corrigeren van bekende fouten heeft zeer hoge prioriteit.

Ad Ib, de documentatie:

Alle code: compiler, driver en run-time systeem, moet zo gedocumenteerd zijn dat een ontwikkelde leek er met redelijke inspanning voldoende uit kan leren om verder het systeem te onderhouden. Ook hier moet alle documentatie door minstens twee handen gegaan zijn.

Ad II, externe soft-ware:

Alle code: compiler, driver en run-time systeem, moet in lopende, overdraagbare vorm aanwezig zijn. Het gebruik van externe soft-ware moet naar redelijkheid beperkt worden, en elk gebruik moet perfect gedocumenteerd worden.

Ad III, documentatie-problemen externe soft-ware:

Door de maatregelen ad II zal de noodzaak tot het gebruik van externe soft-ware afnemen, en daarmee de invloed van slechte documentatie daarvan.

7.3. Het is duidelijk dat het bovenstaande vele malen meer mankracht vergt dan we er op het ogenblik aan besteden. Ik voorzie echter dat wanneer we de hier bereikte conclusies naast ons neer leggen, we uiteindelijk met een schijnbaar wel, maar feitelijk niet overdraagbaar stuk soft-ware komen te zitten. Programmeren is misschien te duur om het goed te doen, maar om het slecht te doen is nog veel duurder.

8. LITERATUUR

[1] *SARA BULLETIN*, 76-2, SARA, Amsterdam, 22 maart 1976.

[2] WIRTH, N., *An Assessment of the Programming Language PASCAL*, International Conference on Reliable Software, SIGPLAN Notices, Vol. 10, Numb. 6 (June 1975), 23.

[3] *PASCAL J Newsletter*, in SIGPLAN Notices, Vol. 11, Numb. 5 (May 1976), 17.

[4] *Control Data Cyber 70 Series Loader Reference Manual*, 60344200 D, blz. 4-9 t/m 4-12.

[5] *Control Data Cyber 70 Series SCOPE Reference Manual*, 60307200 L, blz. 11-30 t/m 11-31.

[6] HEINLEIN, R.A., *Time Enough for Love*, G.P. Putnam's Sons, New York, 242.

[7] *Control Data Cyber 70 Series Record Manager Reference Manual*, 60307300 J, blz. 2-32.