Over trommelpaginatransporten.

1. Om het niet te moeilijk te maken beperken we ons voorlopig tot array-paginas.

1.1. De noodzaak tot transport wordt door het behoeftige programma zelf gedetecteerd en wel in doofheid. Als het nl. niet in doofheidgebeuren zou, zou, na de constatering, dat de pagina er wel is, deze nog onder je vingers weggeratst kunnen worden.

1.2. Als transport noodzakelijk blijkt, dan gaan we naar de coordinator, dwz. het programma komt op de wachtlijst. We keren pas in het programma terug, als de paginatransport voltooid is; we keren dan in doofheid in het programma terug en de aangevraagde arraypagina is geheiligd.

1.2.1. De pagina is geheiligd met de volgende bedoeling: als het transport voltooid is, kán dit programma weer verder; het is niet gezegd, dát het ook onmiddellijk weer verder gaat. In die tussentijd mag de pagina niet verdwijnen!

1.2.2. Het terugkeren in doofheid is niet strikt noodzakelijk, wanneer het programma maar eerst het array-element selecteert en daarna pas de pagina ontheiligt. Keren we terug in doofheid -wat we voor niks krijgen- dan zijn we vrij in de volgorde van elementselectie en ontheiliging. Het programma heeft de plicht de arraypagina te ontheiligen, het element te selecteren (volgorde vrij) en weer horend te worden. Ook dit krijgen we cadeau via de herstellende terugsprong. Een nevenvoordeel is, dat wanneer dit programma verder gaat in nog even doofheid, de ontheiliging gegarandeerd binnen afzienbare tijd na voortzetting plaatsvindt.

1.3. Ik neem aan, dat in geval van noodzakelijk arraypaginatransport de heersende programmapagina ook geheiligd wordt. Dit vergemakkelijkt de terugkeerprocedure aanmerkelijk. Als we dit doen, moet naar ik aanneem ook de programmapagina in doofheid ontheiligd worden (zie 1.1., 1.9.2. e.v.).

1.4. Onbewezen is, dat door de heiligingsprocedure de boel niet vast kan lopen. Dit bewijs kunnen we pas proberen te leveren, als we een volledig overzicht hebben over de aanvraagprocedure bij afwezige (array)pagina.

1.6. Ik beperk me voorlopig tot dat stuk van de coordinator, dat permanent op de kernen aanwezig zal zijn, op een verrekt vaste plaats. De coordinator zal een aantal zg "coordinatorroutines" omvatten; als regel zullen deze met een doofmakende subroutinesprong worden aangeroepen. Het aanroepen van een coordinatorsubroutine op zich zelf betekent niet noodzakelijkerwijze "verlating van het heersende ALGOL-programma". Als regel zal het zo zijn, dat XXX de coordinatorroutine vaststelt, of het XXX heersende programma ongehinderd door kan werken of niet. Zo ja, dan gaat het ALGOL-programma weldra ongehinderd verder -de machine wordt horend bij de herstellende terugsprong uit de coordinatorroutine. De andere mogelijkheid is, dat de coordinatorroutine beslist, dat dit (ALGOL+)programma niet ongehinderd door kan gaan. In dat geval wordt het programma verlaten, dwz. het is niet langer "het heersende programma", het wordt als non-actief op de wachtlijst van de coordinator geplaatst en de coordinator kiest een ander programma en maakt dit heersend.

Ik zoek hier misschien wat spijkers op laag water; wat ik ook probeer is het vinden van een terminologie. In de text van het LAGOL-programma fungeert menig aanroep van een coordinatorroutine als een "mogelijke verlating". Binnenin de coordinatorroutine is de al of niet verlating van het programma een bewuste zaak en kunnen we voor de verlating in doofheid, en na de terugkomst, doelbewuste maatregelen treffen (heiligen en ontheiligen b.v.)

1.7. Coordinatorroutines mogen "over hun break point heen" geen vaste geheugenplaatsen met variabele informatie bezetten: ze kunnen immers parallel geactiveerd worden. Functioneel blijven de betroffen acties dus tot het aanroepende programma behoren. Uit ruimtelijke overwegingen -ook al omdat ze geacht worden zich permanent op de kernen te bevinden- staan ze ook in het geval van multiprogrammering slechts in enkelvoud op de kernen.

1.8. Een gevolg van de graad van bewustheid van een mogelijke onderbreking is de volgende: een gewone interruptie impliceert, dat bij stapelschuiven het B-register aangepast moet worden en dat we heel voorzichtig moeten zijn met absolute verwijzingen naar stapelplaatsen. In een coordinatorroutine kunnen we veel vrijer zijn: na terugkeer onder het break point kunnen we immers testen of de stapel verschoven is en zo ja, wat adressen aanpassen! (dit moet om practische redenen liever niet de spuigaten uitlopen!)

1.9. Voorbeelden van coordinatorroutines zijn de P-operatie, de V-operatie en de indiceeroperaties (schrijvend en lezend). Als het een klein array blijkt te zijn loopt het helemaal met een sisser af!

1.9.1. Opm. De vorige zin sterkt mij, door de moeilijke scheidbaarheid van "run time system" en coordinator in de overtuiging, dat we over multiprogrammering in machinecode voorlopig niet hoeven te denken.

1.9.2. In hoeverre de aanroep van een coordinatorroutine in het geval van break point "heiliging" van de aanroepende pagina impliceert zal er van geval tot geval van afhangen. Iemand zou kunnen stellen, dat in elk programma de heersende pagina altijd heilig is, maar ik voel daar niet zoveel voor, als het betekent, dat elk programma, ongeacht hoelang het op non-actief staat, altijd minstens 1 pagina blift bezetten. Je kunt b.v. kijken, of je ermee volstaan kunt heiliging alleen te introduceren in die gevallen, waarvan je er op aan kunt, dat de inhibitie op korte termijn opgeheven zal zijn. Het wordt nogmaals duidelijk, dat we de "deugdelijkheid" van het concept heiligheid pas kunnen bekijken, als een concreet voorstel op tafel ligt. De rol van Advocatus Diaboli lijkt me in dezen uiterst belangrijk! Het zou me niet verbazen, als ons idee over heiligheid in de loop van deze onderzoekingen drastisch zouden veranderen.

1.10. De coordinator-wachtlijst bevat alle programma's in de machine, onderverdeeld naar twee categorieen, de geblokkeerden en de uitvoerbaren.

Geblokkeerde programma's zijn programma's, die niet verder kunnen, doordat ze op een seinpaal staan te wachten, op de voltooiing van een voor hun voortzetting noodzakelijke gebeurtenis.

Uitvoerbare programma's zijn programma's, die wel verder kunnen, ware het niet, dat de X8 maar 1 van hen verder kan helpen. (Ik neem aan, dat het programma, dat door de X8 uitgevoerd wordt, onder de uitvoerbare gerangschikt blijft; de term "wachtlijst" is dan wat merktwaardig, maar laten we daar maar in berusten.) Als de besturing echt in de coordinator zit, dan is de term in orde.)

Het effect van de P-operatie kan zijn, dat een programma van uitvoerbaar nu geblokkeerd wordt, het effect van de V-operatie kan zijn, dat een (ander) programma uit de groep der geblokkeerden naar de uitvoerbaren wordt overgeheveld. Een programma, dat in actie door een interruptie onderbroken wordt, blijft gerangschikt onder de uitvoerbare!

We kunnen het ook zo zien: als in een programma een P-operatie geinitieerd wordt, dan was op dat moment het programma in actie, dus uitvoerbaar. Nu zijn er twee gevallen. Of de heersende waarden van de betroffen seinpalen vormen geen beletsel, of zij doen dit wel. In het eerste geval worden zij afgelaagd, de P-operatie

is daarmee voltooid en het programma blijft uitvoerbaar. (Of het ook in actie blijft is een heel ander chapiter!) In het tweede geval wordt het programma uit de groep der uitvoerbaren gehaald. De groep der geblokkeerden bevat dus alleen alle programmas, die in een geinitialiseerde, maar niet voltooide P-operatie zijn blijven hangen.

1.11. Ten aanzien van de uitvoerbare programma's kunnen we twee wegen inslaan inzake heiliging. We beschouwen een programma, dat in actie was, door een interruptie is onderbroken en tengevolge daarvan op het uitvoerbare gedeelte van de wachtlijst is gekomen.

1.11.1. In de ene techniek staan we toe, dat de programma-pagina, waarmee we op het moment van interruptie mee bezig waren tijdens het wachten verdwijnt. Dit impliceert:
a) dat we in de wachtlijst bij de status quo gegevens het terugkeeradres invariant moeten opbergen.
b) dat als de coordinator er toe besluit, dat deze uitvoerbare er nu aan toe is, om voortgezet te worden, er getest moet worden, of de programmapagina, waar we vandaan kwamen en waarheen we weer terugwillen, nog op dezelfde plaats in de kernen zit, maw. de voortzetting moet gespeeld worden als een goto-statement naar een andere, dwz. mogelijk niet aanwezige pagina.
c) dat dan in het geval van afwezigheid door de nu aangevraagde trommeltransport het programma weer op de lijst van de geblokkeerde programmas voorkomt.

Als nu na voltooid transport dit programma weer bij de uitvoerbare terecht komt, maar niet direct aan bod komt en dus de pagina voor feitelijk gebruik alweer verdwenen kan zijn, dan is dit niet alleen triest, maar ook gevaarlijk: dit zou nl. betekenen, dat we de mogelijkheid van "effectloze" trommeltransporten geintroduceerd hebben. Zijn we er dan nog zeker van, dat we niet het "Na U" effect geintroduceerd hebben? Dit zou te ondervangen zijn door de kernpagina op het moment van beslissing, dat het transport zal plaatsvinden te heiligen en de ontheiliging pas plaats te laten vinden door de feitelijke voortzetting van dit programma.

1.11.2. In de andere techniek gaan we er van uit, dat het terugkeeradres van een door interruptie onderbroken programma als physisch kernadres opgeborgen wordt, maar dat tijdens uitvoerbaar wachten de pagina in kwestie altijd heilig zal zijn. De andere manier, waarop een programma bij de uitvoerbaren kan komen is via de geblokkeerden. Als blokkade altijd vanuit een coordinatorroutine plaats vindt, dan mogen we ook met een physisch adres volstaan. De coordinatorroutine is immers permanent op de kernen; de coordinatorroutine krijgt dan de plicht om er in geval van gebruikt break point voor te zorgen, dat hij zijn schepen niet achter zich verbrandt.(Door hetzij het terugkeeradres invariant te bergen, dan wel over het break point heen de aanroepende pagina te heiligen.)

Deze techniek kan in eerste instantie op twee wijzen gespeeld worden. We kunnen de heersende pagina altijd heiligen, dit vereist maatregelen bij elke sprong van de ene pagina naar de andere. Een coordinatorroutine heeft dan mogelijk de plicht om bij gebruikt break point -als het te lang kan durende heiliging op te heffen en na terugkeer testend terug te springen.

De andere manier is, dat we bij goto statements naar andere pagina's helemaal niets doen, maar in het geval van een interruptie de verlaten pagina heiligen en bij voortzetting ontheiligen. In dit geval moeten we in een coordinatorroutine mogelijk -als de break gegarandeerd kort duurt- deze pagina heiligen voor de duur van de break.

1.11.3. Door de bibliotheek te behandelen zoals Voorhoeve gesuggereerd heeft, kunnen we niet volstaan met in de KIE een heiligbit op te nemen. Immers: vanuit

een bibliotheekpagina wordt een coordinatorroutine aangeroepen en de bibliotheekpagina wordt heilig achtergelaten. In de tussentijd komt een ander programma aan bod, dat eveneens deze bibliotheekpagina via een dergelijke coordinatorroutine verlaat. Een van de beide programma's wordt het eerste weer voortgezet: dit programma mag dan de bibliotheekpagina niet ontheiligen! Toen ik dit ~~xxxxxxxxxxxx~~ addertje vanonder het gras had losgewoeld, was ik wel een beetje tevreden met mezelf!

De remedie is duidelijk: de Kie heeft inplaats van een heiligbit een heiligtelling. Telling = 0 betekent profaan, telling positief beteknt heilig. Heiligen betekent telling met 1 verhogen, ontheiligen betekent telling met 1 verlagen. Het welbekende "Heilig, heilig, driemaal heilig" begint een bijzonder relief te krijgen.

Opm. Moeten we oer rpogramma een "ontheiligplicht bijhouden voor het geval het programma voortijdig beeindigd wordt? Ik denk van wel.

18 december 1963

E.W.Dijkstra