Eerste verkenning over de dood van programma's.

1. De processen, die door de CM's worden uitgevoerd, zijn onsterfelijk. Dit rapportje handelt dus duidelijk over een PM-aangelegenheid.

2. Een PM heeft een onsterfelijk buitenste blok, het PM-blok. Dit wordt nooit verlaten, het is in dit blok, dat de in- en uitvoerstromen thuishoren; evenzo hoort hier thuis de statusinteger "runnumber". De PM bevindt zich in dit blok, wanneer op standaardband gewacht wordt. (Deze toestand, ook in een statusvariabele voor de bandlezerselectie vastgelegd, zou ook in "runnumber = 0" kunnen worden vastgelegd.) Zodra een vermeende standaardband wordt gevonden, wordt het runnumber ingevuld; als het een algolprogramma blijkt wordt op grond van de buiten het programma vermelde pluncherbehoefte de acceptabiliteit getest; zo ja, dan gaat de PM een binnenblok in, waarin de globalen voor de vertaler gedeclareerd worden, wordt de vertaler als procedure aangeroepen en als we daaruit terugkomen (en het programma was goed vertaald) dan roepen we het programma aan. Komen we hier helemaal legitiem uit terug, dan doet de standaardtext van de PM de blokexit van het ingegane blok, we zijn terug op het niveau van het PM-blok. De eigen informatie (segmenten, heiligingen etc.) van het algolprogramma zijn hiermee afgehandeld, de PM hoeft niet meer te doen, dan de in- en uitvoerstromen tot hun nuchtere staat terug te brengen. Als er nog iets in een invoerstroom zit (of zit te komen), dan moet "skip until end of tape" gedaan worden (inclusief "geen interesse meer, zie de tape reader CM), als in een pluncherstroom een unfinished document zit, moet dit worden voltooid en aangehaakt, als in de printerstroom nog een unfinished form zit (hier zit een WC-rol conventie: ik neem aan, dat deze er in zit, zo er ueberhaupt van de printer gebruik gemaakt is) wordt deze als de laatste gemarkeerd en "number of final documents nfd" wordt met 1tje opgehoogd. (Dit even met Coen opnemen.)

Nadat alle stromen zijn afgemaakt zet de PM zijn behoeften op 0 (inclusief repercussies voor verdachte plunchers), zet het runnumber op 0 en gaat op standaardband staan wachten.

Dit is, zoals ik me de natuurlijke dood van een programma voorstelde. In verband met schijndood en de gewelddadige dood (zie onder) komt er nog wel wat bij.

3. De operateur heeft de mogelijkheid (waarvoor moge de hemel weten) om een programma schijndood te verklaren. Het moet dan op een zacht pitje zitten wachten, totdat het weer, via de console, wordt gewekt.

Het schijndood maken is een bevel, gericht tot een programma in een PM; als er geentje in zit (runnumber = 0) dan lijkt het commando me Non Applicable. Ook als het programma, dat er in deze PM zit, al schijndood is. Waarschijnlijk zullen we voor een aanwezig programma ook de boolean "op sterven" hebben; dan moet er ook maar gestorven worden en ik dacht, dat het bevel tot schijndood ook dan NA moest wezen.

De complicatie bij schijndood is, dat het programma dat zachte pitje bij voorkeur wel plaatst op een onschuldig punt. Wij hadden zo gedacht "niet in een roode sectie". Als een programma wit en uitvoerbaar is, dan kan het commando tot schijndood meteen verwezenlijkt worden: men boekt het als wachtend op een voor dit doel in elke PM voorkomende seinpaal. Anders moeten we alleen het verzoek tot schijndood boeken (weer een toestand, waarin het commando "val schijndood als NA behandeld zou kunnen worden).

Deze blokkering van schijndood moet achterwege blijven, als de PM rood is (hij moet nog doorwerken) en is niet lekker codeerbaar, als de PM al op een andere seinpaal staat te wachten. De uitgestelde schijndoodheid moet effectief worden bij de witmakende V-operatie en de voltooide P-operatie. (Opmerking van Piet: de schijndood hoeft pas effectief te worden als je dreigt de controle aan een wit proces metterdaad te gunnen. Dit geldt ongeacht de reden tot uitstel en zou tot compactere codering aanleiding kunnen geven.)

De wekoperatie is nu makkelijk. Als de schijndood gevraagd, maar nog niet gerealiseerd is, kan worden volstaan de notitie weg te nemen. Als de aanvraag niet hangt, moet de schijndood gerealiseerd zijn (anders NA) en de message interpreter kan op de daartoe strekkende seinpaal een V-operatie uitvoeren. (Let wel, dat het veilig lijkt om voor de schijndood een specifieke seinpaal per PM in te voeren!)

4. De gewelddadige dood komt vanuit een andere AM (hetzij van een CM, die ontdekt, dat een ponser kapot is, hetzij via de message interpreter van de operateur, misschien ook van de PM zelf, die ontdekt, dat hij "onmogelijk verder kan". Ik zie dit nog niet zo direct, zo mogelijk openhouden.)

Het grote probleem van de gewelddadige dood is, dat we dit programma moeten beeindigen, dwz. er een punt aan moeten draaien, zodat het toch nog well-behaved blijft. Een eerste vereiste is, dat het in elk geval zover doorgaat (zo nogig), dat ook de toestand "schijndood" zou mogen ingaan. Dit geeft ons een nieuw criterium om vast te stellen, waar roodheid begint en waar roodheid moet ophouden: buiten roodheid moet de PM in een hanteerbare toestand achterblijven. Nadat we nu straks hebben laten zien, wat we ons bij dat hanteren voorstellen, moeten we de stukken PM-programma, waarin roodheid voorkomt en die we inmiddels gemaakt hebben, in dit licht nog eens even duchtig nakijken.

Er is een communicatieapparaat, dat hier wat roet in het eten doet, en dat is de trommel. Ik wil een programma liever niet vermoorden, als het nog op een seinpaal staat te wachten, in het bijzonder niet, wanneer het aan de segment controller een segment gevraagd heeft. Als we af kunnen spreken, dat sterven niet begint, zolang de PM nog op een seinpaal wacht, dan kunnen we dit als volgt spelen (NB. Het aanvragen van een segment aan de segment controller gaat buiten alle roodheid om: het kan in roodheid, het kan in witheid). Bij de opdracht aan de segment controler kan de PM een boolean "Aanvraag Segment" true zetten.

De structuur in de PM wordt dan

```
AS := true
   aanvrage van segment
   V(SCS)
AS := false ; P(eigen segment seinpaal)
```

De laatste twee statement staan op een regel, om heel nadrukkelijk af te spreken, dat ze in doofheid na elkaar gebeuren. Nu spreken we af, dat de initialisering van de zelfmoordriten niet plaats kan vinden, als AS. Hiermee bereiken we, dat de aangevraagde pagina's er zijn, voordat ze wellicht vernietigd zijn. (Ik schreef "pagina's, ik bedoelde "segmenten".)

Zo te zien, kunnen we met 1 variabele "killvar" al een heleboel doen. Tentatief zou ik deze de volgende waarden met de volgende betekenis willen geven (De op pag.0 gesuggereerde codering van "runnummer = 0" kan dan vervallen.)

killvar = 0 De PM is leeg
killvar = 1 De PM is bezig met een programma, dat volslagen levend is
killvar = 2 Er is aanvrage tot schijndood
killvar = 3 Het programma is schijndood (hangt op de seinpaal killsem)
killvar = 4 Er is aanvrage tot zelfmoord
killvar = 5 Het programma is tot de zelfmoordriten overgegaan.

(Of de boolean AS hier ook in gecodeerd moet worden, bv. door killvar + 8 is voor mij nog een open vraag.)

Ik stel me nu op hetstandpunt, dat het agens, dat de zelfmoordopdracht geeft, hiervan via de consoleteleprinter melding maakt, maar dat we de PM (of liever: het er in zittende programma) niet de gelegenheid geven, nog een brief aan vrienden en bekenden ten afscheid te schrijven, dus geen rapportering over de status, waarin het programma zich bevondt, toen tot de zelfmoordriten werd overgegaan. (Hoogstens voeren we nog in

killvar = 6 Het programma is op eigeninitiatief tot de zelfmoordriten overgegaan, in tegenstelling tot killvar = 5. Op de laatste printerform kan dan nog een kreet komen te staan "ik heb mezelf beeindigd, dan wel "ik ben er afgetrapt".)

Ik hoopte, dat we de overgang tot de zelfmoordriten geëffectueerd zou kunnen worden door in de status van het onderbroken programma zo te knoeien, dat zo ongeveer met het mechanisme van de General Goto de besturing springt naar het punt in het PM-blok, waar XXX de PM de stromen gaat afmaken. Dit is niet genoeg, want er kunnen vanwege de SIS twee programmasegmenten heilig zijn -en de general goto ontheiligt er maar 1tje, als de moord plaatsvindt wanneer een arraysegment heilig is aangevraagd, dan zitten we daar ook nog mee. (De moord wordt wel uitgesteld, totdat dit segment gearriveerd is). Ik kan me voorstellen, dat een en ander bij het heilig aanvragen van arraysegmenten een extra notitie vergt en dat de overgang tot de zelfmoordriten zo min mogelijk door de coordinator en zo veel mogelijk door de PM zelf XXXXXXXX gebeurt. Het wegwerken van een stukje SIS (ontheiliging en zo) is nl. een handeling, die bij de dynamische fout onder auspicieen van de PM zelf zal gebeuren.

5. De dynamische fout.

Na ampele overwegingen zijn we tot de conclusie gekomen, dat de reactie op een dynamische fout een zuivere PM-aangelegenheid is. Ik stel me zo voor, dat deze alleen tijdens killvar = 1 zal kunnen voorkomen (in rode stukken wil ik helemaal niet zulke griezelige dingen kunnen doen). De PM zal dan de stapel moeten afbreken totaan de eerste Working ÑÑ Space Pointer; als de fout in een SIS optreedt, dan moet hij dit door stapelinspectie ontdekken, als er nog een arraysegment heilig is, dan weet het foutendetecterende mechanisme dat, en kan XXX kan dat arraysegment ontheiligd worden. Is de stapel teruggebracht tot het eerste niveau, waarop SE1 aangeroepen kan worden, dan zal dat moeten gebeuren om een aanroep van de procedure ALARM te kunnen simuleren. Met de sprong daarnaar toe wordt de laatste programmapagina ontheiligd, ALARM kan onder zich de stapel zoveel inspecteren al hij wil en kan tenslotte springen naar het punt in het PM-blok, waar we na programmabeeindiging terecht komen.

De microscopie van het mieren aan de status onder in de HAM-kamer en het toespelen naar een tussengeschoven systementry is een spelletje, dat me wel boeit; ik beschik zelf niet over de kennis om direct te zien, hoe dit moet, ik dacht wel, dat het kon en wil er graag over helpen denken.

EWD.