A sequel to EWD126.

I recall from EWD126 the definitions:

p = the total number of segments that are part of the PM's
i = the total number of segments that are part of the input streams
o = the total number of segments that are part of the output streams.

The vital inequalities to be respected are:

1) $i + o + p \leq tot$
2) $i + o \leq$ transput area
3) $i + p \leq tot - reso$
4) $i \leq$ transput area $- reso$.

To guarantee guarantee these inequalities, we introduce the "differences":

fiop	(initially = tot)	say: 1024 - something
fio	(initially = transput area)	say: 256
fip	(initially = tot - reso)	say: 960 - something
fi	(initially = transput area - reso)	say: 192

The operations that performs changes in i, o and/or p do so by changing the value of the four quantities listed:

SH1	p-increase:	"dec(fip)" and "dec(fiop)"
SH2	p-decrease:	"inc(dip)" and "inc(fiop)"
SH3	i-p transition:	"inc(fio)" and "inc(fi)"
SH4	p-o transition:	"dec(fio)" and "inc(fip)"
SH5	i-increase:	"dec(fiop)", "dec(fio)", "dec(fip)" and "dec(fi)"
SH6	o-decrease:	"inc(fiop)" and "inc(fio)".

Sequencing of the regions of program will be done in such a way, that the four quantities listed never get negative values.

Before going on, we shall first justify the inequalities, then the four quantities.

The first inequality is obvious: it expresses the maximum size of the drum.

The second inequality expresses the existence of transput area: this has been introduced because the SV's of (possible) transput segments will impose a permanent core store occupation.

The third and fourth inequality express that part of transput area (viz reso) is reserved for output exclusively. This is a selfimposed restriction which we exploit in the following way.

If the third or fourth inequality is in danger of being violated (i.e. when fip or fi is in danger of becoming nebative) we signal "disaster" (viz. store exhaustion). Disaster detection must be implemented anyway, we shall see below, that this convention pins down the disaster detection to a few well defined points. The couterpart of this disaster detection is, that when at these few points no disaster has been detected, we must be able to guarantee that the processes will be able to continue. Or more precisely, when the operations "dec(fi)" and "dec(fip)" occur only at SH1 (in a PM) and at SH5 (in a CM for input). These are the only

points in which disaster can be signalled. If this disaster has not been signalled, dec(fiop) and dec(fio) might force a delay (viz. when fiop = 0 or fio = 0). In order that in due time the desired i- or p-increase indeeed becomes possible, we must see to it, that the contents of the output streams are constantly such that, if necessary, the state "o ≤ reso" can be reached without relying on further PM-or input CM activity.

It is necessary that we can get o ≤ reso, in order to be able to effect the i- or p-increase that has been requested without disaster signalling, but this condition is insufficient. For it means that upon the total size of the output streams, the uppre bound reso may be imposed: if now all reso segments, available for output buffering, are occupied by output for which at this moment no equipment is available, we are still stuck, as we lack the room to keep the "moving streams" going.

We therefore impose upon the so-called "passive output" (for the definition, see below) a lower upper bound, called "maxu", where reso = maxu = maxpor (say =4), the maximum size of a portion of information to be put in transput area at SH4 and taken from it at SH6. Thus we gaarantee that, if necessary
a) all but reso segments become available for i and p
b) the remaining reso segments are sufficient to keep the output going.

Remark. The active output may be confined to maxpor segments in transput area. If more than one output CM has to get its food via this narrow buffer, we cannot expect them to run at maximum speed. We leave it at that: we could have remedied this -without altering maxu- by increasing reso, i.e. by being quicker in signalling disaster, but there is nothing to be gained from that.

We make a second use of the boundary maxu: if "maxu < o", there is not only active output possible, but we even guanatee active output. This has the subsidiary advantage, that in SH1 -a frequent occurrence- we have only to test, whether the desired p-increase is possible. If not, and relieve has to come from active output, there is no need to inspect, whether now the moment has come to activate: output is already going and the PM has only to wait. The second advantage is that SH2 (i.e. p-decrease), also a frequent occurrence, is never an occurrence upon which other processes may be waiting for their relieve.

So much for the inequalities. Now the four quantities.

They have been introduced for two reasons: the minor reason is that they enable us to test for danger situations at a zero value, instead of comparing with some bound. The main reason, however, is the following. Note, that we have carefully avoided to define "fiop = tot - i - o - p" etc., although their initial values and the dec- and inc-operations are consistent with these definitions: these definitions would only be correct, if in SH1, say, "dec(fip)" and "dec(fiop)" were performed simultaneously. But this is not the case!
In SH1, disaster is signalled if fip = 0, otherwise, dec(fip) is performed. After that, fiop = 0 is tested, if <u>false</u>, dec(fiop) is done immediately, if <u>true</u>, however, the PM has to be delayed, the p-increase has to be postponed. (Without the "predecrease" of fip, we should have to face the situation, that also the completion of a delayed p-increase could generate the disaster condition.)

Now we come to the definiton of passive output,i.e. output in the buffer, of which we are not sure whether we can get rid of it (or even: are sure that we cannot get rid of it) without further PM-activity. It is closely connected to the banker's algorithm and only applies to documents for the plunchers,

(Use of the printer is not subjected to the banker's aporoval; accordingly, printer forms in the buffer never contribute to the passive output. We recall that the PM's offer the printer information in SH4 form wise and that we allow the printer CM to change information stream after each form, if necessary. We call a from inmage in a printer stream "a (printer) portion".)

For the pluncher streams, however, the PM's produce so-called "documents" of a priori unknown length. They attach (in SH4) then information portion wise, here a portion being a segment. The concept of "a document" is a restriction for the plunchers. As soon as a pluncher has selected a document (i.e. irrevocably decided that now the output of a certain document will be initiated), it will not become available for any other purpose until the document in question has been processed completely. If a pluncher selects a finished document, it sets itself to an already well defined task: the document is in its entirety available in the output buffer and the pluncher can go on at full speed.

The situation is drastically changed, however, if a pluncher selects an unfinished document, for then it becomes tied up until the PM will finish the document. The critical situation in which a pluncher has selected an unfinished document may only arise under the banker's approval. We describe it as the pluncher being tied up to the PM, or rather in the banker's terminology "being borrowed by the PM". The PM starts the loan of a pluncher at the moment that the pluncher selects an unfinished document, it ends the loan at SH8 "Finish Pluncher Document" (This is a PM-action). If the document has been finished before it has been selected, there is no loan involved and the banker's algorithm is not applied.

We introduce, for the banker, but also for the definition of passive output

plocash = the number of plotters that are not borrowed
puncash = the number of punches that are not borrowed.

These two quantities tell us "How much the banker has in cash". (Mind that "unborrowed" does not mean "idel": a pluncher engaged in the processing of a finished document is unborrowed, but certainly not idle!)

To the passive output belong

a) all unfinished, unselected pluncher documents

b) all finished unselected plotter documents if plocash = 0 and all finished unselected punch documents if puncash = 0.

Catagory a) represents the information of which we are not sure, whether we can get rid of it via the output (for this would imply selection of the unfinished document, an action which is subjected to the banker's approval); catagory b) represents the information of which we are sure that we cannot get rid of it with stopped PM's: the banker has no corresponding equipment in cash, i.e. the corresponding equipment has all been borrowed.

The total size of the passive output may not exceed maxu, growth of the passive output must therefore be carefully watched. We propose to restrict the growth of passive output to increasing the length (in SH4) of a document in catagory a) and rule out that passive output increases, due to the fact that finished documents, first outside catagory B0 come to fall inside catagory b), due to "cash" going from 1 to 0.

For the single plotter this just means "never select an unfinished document if there is still a finished plotter document"; for the (more) punches this means "if you are the last unborrowed punch, then never select an unfinished document in the presence of a finished one." (The rule is certainly obeyed, if also punches only select an unfinished document, when no finished ones are available, as we may program them.)

To keep track of passive output we introduce for each pluncher stream:

lufd (=length of unfinished, unselected document). This quantity gives the length (in segments) of the document under construction, provided it is unselected. Otherwise, lufd = 0.

The contribution to passive output of catagory a) is therefore SIGMA(lufd), taken over all pluncher streams.

For the contribution to passive output of catagory b) we introduce for finished plotter documents "plofp" and for finished punch documents "punfp" (...fp : finished passive documents).
Naturally: 0 < plocash implies plofp = 0 and
0 < puncahs implies punfp = 0.

Passive output will be equal to

SIGMA(lufd) + plofp + punfp

and as this may not exceed the upper bound maxu we introduce again the difference, the number of segments free for passive output

fpas (initially = maxu)

a quantity which is not allowed to have a negative value.

In order to describe the modification of the quantities introduced we mention the variable (per pluncher stream)

unfsel =1 means: the unfinished document in this stream has been selected (i.e. the corresponding PM has borrowed the corresponding output CM)
=0 otherwise.

In SH4: (p-o transition)

```
if unfsel = 0 then begin inc(lufd); dec(fpas) end;   dec(fio)
```

Thus, if the document is unselected (catagory a) the corresponding changes in passive output control are booked.

In SH7: (document selection) in the case of selection of an unfinished document (under banker' approval!):

```
dec(...cash); INC(fpas, lufd); lufd:= 0; unfsel:= 1
```

The document selected leaves passive output from catagory a. No finished documents shall enter passive output in catagory b) (see previous remark).

In SH8 (Finish Plucher Document)

```
if unfsel = 1 then
begin unfsel:= 0; XXXXX inc(...cash);
      if ...cash = 1 and XXXXX 0 < ...cash then
      begin INC(fpas,...fp); ...fp:= 0 end
end
              else
begin if ...cash = 0 then INC(...fp,lufd) else INC(fpas, lufd);
      lufd+= 0
end
```

If a selected document is finished, this ends a loan; if now cash changes from 0 to 1, there may be documents in catagory b) now leaving passive output. If an unselected document (by definition in catagory a) is finished, it either goes to catagory b or it leaves passive output.

This far we have described the clerical variables and shown how we can operate upon them, consistently with their meaning. Now we shall describe the structure of the programs that together see to it, that the five basic quantities that must remain non-negative, indeed remain non-negative. For the present moment we restrict ourselves to the PM's and the output CM's.

Each PM has a private semaphore, called "pmsem" and a private state variable called "pmshr". In their homing position, they have both the value "0". Analogously, each output CM has a private semaphore, called "ocmsem" and a private state variable, called "plovar" for a plotter,"punvar"for a punch and "privar" for a printer. (If we wish to talk about the statevariables for the combined plunchers, we may call them "pluvar"). Also here, the homing position is = 0.

Over these we have a universal, binary semaphore, called "SHS", which caters for the mutual exclusion of the inspection and modification of the common clerical variables.

The easiest operation is SH2: p-decrease. If the number of non-empty segments killed equals "killed", the program runs:

```
SH2: P(SHS); INC(fip, killed); INC(fiop, killed); V(SHS)  .
```

(Note: in the present description we shall not investigate, whether in trivial cases the P(SHS) and V(SHS) can be omitted and replaced by disabling and abling the interrupts.)

The next easiest operation is SH1: P-increase. In a PM this always occurs for a single segment. Here fip and fiop have to be decreased both, but this is done in two steps. The program is as follows:

```
P(SHS);
if fip = 0 then disaster else dec(fip);
if fiop = 0 then
      begin pmshr:= 3; V(SHS); XXXXXXX P(pmsem) end
            else
      begin dec(fiop); V(SHS) end
```

after which the PM has got the permission to fill an empty segment.

The PM tries to do dec(fip) and then XX dec(fiop); if the first one is unsucessful, disaster is signalled, otherwise it is done. If the second one is

also successful, we are immediately ready, otherwise the PM leaves the critical section after having set

pmshr = 3 "The PM is waiting on account of inability to perform dec(fiop) in SH1"

After having left the critical section it waits.

The complementary action takes place in the critical section of a process that has increased fiop. After having done so it has to inspect, whether any pmshr = 3 is present. If so, it performs for the PM concerned

"dec(fiop); pmshr:= 0; V(pmsem)"

i.e. the decrease, which the PM was unable to do; then it sets pmshr = 0, as the decrease requested has been performed and finally it signals to the waiting PM that it can go on.

Note. if "fiop" is the limiting factor, total output must exceed reso (otherwise, namely, disaster would have been signalled). On the other hand we shall guarantee that there will be active output already, if total output exceeds maxu. Therefore: 1) the task to reslove "pmshr =3" can be delegated to 0-decrease in active output only and 2) the PM has nothing else to do because there will be active output and the o-decrease will occur.

The other operation with possibly two decreases is SH4: p-o-transition in the case of an unselected document for a pluncher. If so, the two decreases will again be a tw stage affair. We give the raw outline of the program, the discussion of "desire" (a stream variable), "banker" (a universal procedure) the constant "maxpor" and the PM-variables "bsn" and "request" will be postponed.

```
SH4: P(SHS); bsn:= "stream identification";
     if unfsel = 0 then
     begin if fpas = 0 then
           begin pmshr:= 1; desire:= 1; banker;
                 V(SHS); P(pmsem); P(SHS) end
                      else
           begin inc(lufd); dec(fpas) end
     end;
     if fio < maxpor then
     begin pmshr:= 2; request:= 1;
           V(SHS);P(pmsem); P(SHS) end
                    else
     dec(fio);
     "Now the p-o-XXXXXXXXX transition can be effectuated, followed by:"
     inc(fip).
```

The quantity desire is a stream variable, its usual value is zero. For a pluncher stream its sole purpose is to influence the procedure "banker" which compares the lengths of various unfinished, unselected documents, but it then takes

"lufd + desire" .

The main point is the distinction between desire = 0 and desire = 1, both with lufd = 0: the difference between a document being empty for lack of interest or for lack of space. In the first case it is not a suitable candidate for selection, in the second case it might be.

The next state, described by the state variable pmshr is

pmshr = 1 The PM is waiting on account of inability to perform dec(fpas) in SH4.

It implies that the blocking stream is a pluncher stream, containing an unselected document. The existence of one or more "pmshr = 1" implies "fpas = 0".

There are two ways, in which the PM can be helped over this barrier

a) After increase of fpas. The processes increasing fpas have the duty to investigate whether now PM's sleeping have to be woken up. Waking up then consists (within a section, critical to SHS)

"pmshr:= 0; dec(fpas); inc(lufd); desire:= 0; V(pmsem)"

(Here the initial choice is the waiting PM; a variable bsn (blocking stream number) tells, in which stream "lufd" and "desire" have to be selected. "bsn" is superfluous: it points to the one and only pluncher stream of this PM with "desire = 1")

b) By selection of the stream in question. for this makes the desired "dec(fpas)" unnecessary. The the cations comprise:

(for the selection) "INC(fpas, lufd); lufd:= 0; unfsel:= 1"

and, provided desire = 1 (then pmshr = 1 as well)
for waking up) "pmshr:= 0; desire:= 0; V(pmsem)"

When the first barrier has been passed, the second comes: instead of asking whether the current value of fio allows the decrease wished for this time, we always decide on

$$fio < maxpor$$

following a suggestion by C.Ligtmans.

The price paid is that the last maxpor-1 available output segments remain unused, when the printer is unused. The advantages, however, become apparent, as soon as we investigate waht happens when plunchers decide on "fio < 1" and only the printer on "fiop < maxpor". In that case output limitation might easily lead to stopping of the printer stream. Also, we can now state, that the existence of one or more *pmshr = 2" implies "fio < maxpor"; if fio increases we can select the PM with pmshr = 2, regardless the type of the blocking stream.

We have the meaning

pmshr = 2 "The PM is waiting on account of the inability to perform DEC(fio, request) for request $\leq$ maxpor.

If some process decides to wake up a PM with pmshr = 2, it does so by

"pmshr:= 0; DEC(fio, request); request:= 0; V(pmsem)"

It is the function of the procedure "banker" to try to wake up PM's with pmshr = 1. It will always be called from within a critical section, it may be called from a PM, but also from a CM.

Its total action may end in two ways:
either there are no pmshr = 1 any more (fpas may have any value) or there are still pmshr = 1 (and fpas = 0) but it has not been able to remedy the situation. This last thing can happen in two ways

a) an unfinished document may be unselectable on account of absence of
an adle plucher of the correct type
b) an unfinished document may be unselectable on accout of disapproval of the banker's algorithm.

Waking up a PM with pmshr = 1 will imply "pmshr:= 0; V(pmsem)", waking up an idle plucher (characterized by "pluvar = 1") will imply "pluvar:= 0; V(ocmsem)". Note that waking up may be applied to the abstract machine (either PM or CM) from which the banker happens to be called.

The only tool available for the banker is to make an unfinished document selected: this may have a double effect.
1) The stream in question may be a blocking stream for a PM with pmshr = 1; this makes the requested dec(fpas) no longer necessary and the PM has to be woken up
2) furthermore fpas may increase on account of which pmshr = 1 can be removed.

Roughly, the procedure "banker has the following body:

```
L0:  if fpas > 0 then goto finish;
     if non exist(pmshr = 1) then goto finish;
L1:  if non exist(pluvar = 1) then goto finish;
L3:  if "there exists an unfinished document in an otherwise empty stream
     with lufd + desire > 0 and a corresponding pluvar = 1, selection of which
     is allowdd by the banker's algorithm" then
select the document:
     begin dec(plucash);
           pluvar:= 0; etc. for plucher; V(ocmsem);
           INC(fpas, lufd); lufd:= 0; unfsel:= 1;
           if desire = 1 then
           begin comment a blocking stream happens to be selected;
                 pmshr:= 0; desire:= 0; V(pmsem) end;
           use fpas for waking; goto L0
     end
finish:

with the procedure use fpas for waking;
     begin LL: if fpas > 0 and exist )pmshr = 1) then
           begin pmshr:= 0; dec(fpas) inc(lufd); desire:= 0; V(pmsem);
                 goto LL end
     end
```

In the conditional clause labelled "L3" the banker inspects the streams with "nfd = 0" (nfd = number of finished documents. "nfd = 0" means that the stream contains at most an unfinished document) for which a corresponding pluncher is idle (pluvar = 1) and inspects them in order of decreasing "lufd + desire",. The first allowed by the banker's algorithm but still with "lufd + desire > 0" is taken. As a rule the banker will select at most one unfinished document: as it looks for the longest one it will cause the maximum increase of fpas possible. Note, that in SH4 the banker is called after the PM has set pmshr and desire: it may be this very document, that needs to be selected.

Further the banker is called whenever a change in circumstances occurs, that may create the possibility for succesful action.

XXXXX A more complete version of SH8 (Finish Pluncher Document) is therefore:

```
SH8:
P(SH5); inc(nfd);
if unfsel = 1 then
begin unfsel:= 0; inc(...cash);
     if ...cash = 1 and ...fp < 0 then
     begin INC(fpas,...fp);...fp:= 0; use fpas for waking end;
     banker
end
               else
begin if...cash = 0 then
     begin INC(...fp,lufd); lufd:= 0 end
                    else
     begin INC(fpas,lufd); lufd:= 0; use fpas for waking;
           if exist(...var = 1) then
           begin...var:= 0; etc. for pluncher; V(ocmsem) end
     end
end;
V(strsem); V(SH5)
```

Remark 1. "strsem" is a streamsemaphore that is also increased at ever new portion added to the stream. It is used to synchronize the output CM, when it has selected this stream.

Remark 2. It is felt that the choice of the PM in the case of multiple "pmshr = 1" in "use fpas for waking" is highly irrelevant.

When the pluncher has finished the processing of a document, it will do a final "P(strsem) XXXXXX -as the complement of the X V(strsem) in SH8- in order to guarantee that the pluncher will never look for a new job, before the corresponding SH8 is duly completed.

If it then finds a finished document on top of one of its (unselected) streams, it takes it, otherwise it does

"pluvar:= 1; banker; V(SH5); P(ocmsem)"

When this last P-operation has been completed, a new document (finished or not) has been selected.

We shall now give a short description of the high speed printer. It may be used from wise, so we are independent of the banker's algorithm. Nevertheless it is unattractive to have the printer forms from different sources all mixed up: it should like to print a number of successive froms from the same program successively. Therefore: if the different printer streams cantain very few forms, the printer will delay selection, if possible, in order to wait, until one of the streams has a reasonable amount available. This delay is no longer necessary, if one of the programs has produced its last from, it is no longer possible , if the total output increases beyond maxu, without other output being active.(Below, we shall use an other criterion to regulate printer activity.)

In each printer stream the final form, produced by a program is marked as such; the number of such final forms is counted (we may use "nfd" for this purpose, now meaning "number of final documents"). It is increased at the chaining on of the final form (by the PM), it is decreased by the printer CM, at processing of the final form.

The printer may be idle in two different ways:

privar = 1 dependent on past history: the last form printed was not a final document, the stream in question, however, is empty, and the printer prefers to wait -as long as possible- for the next from from the same stream

privar = 2 independent of past history: the printer has not printed yet, or the last from printed was a final document (or one program is in danger of monopolizing the printer).

We must prevent one very active PM from monopolizing the printer; we propose to do so, by "forgetting the past history" after, say 20 consecutive forms from the same stream.

Idleness of the printer can always be removed for two different reasons
1) there is a form available that the printer should love to select
2) there is no such attractive form, but the printer should get in action on account of the maxu barrier.

In the case of "privar = 1" the attractive reason is a form in the current stream, in the case of privar = 2, the attractive reason is, in some printer stream, a positive nfd.

The unattractive reason is- under both circumstances- something of the nature of total output exceeding maxu, and no other output active. Let us construct this criterion a little bit more carefully.

For the pluncheS we have passive output and the active output. We can leave the active output out of consideration, because that will disappear: without further PM activity, the passive output will remain, and the printer activity must be such, that yet the total amount of buffered output will in due time not exceed maxu. As fpas is the amount of growth of passive information, that we can admit, the criterion for activity of the printer is

fpas < sump

where "sump" is the sum of the lengths of the printer streams (measured in segments).

The quantity "sump" is increased as part of the SH4 for a printer document and it is decreased as part of the printer form processing by the printer CM.

Similar to "banker", that caters for pluncher document selection, we have a procedure "printer selection"; it will only be called from within section, critical with respect to SHS and will have no effect, when the printer is not idle. First it looks for attractive reasons to go on printing, then it looks for the unattractive ones.

```
printer selection:
if privar = 1 then
begin if current printer stream non empty then
      begin privar:= 0; V(prisem) end
                                        else
      begin if fpas < sump then privar:= 2 end
end ;
if privar = 2 then
begin if exist (0 < nfd) then
     begin privar:= 0; select stream with positive nfd etc; V(prisem) end
                        else
```

```
    begin if fpas < sump then
        begin select stream with printer information etc; V(prisem) end
    end
end
```

This routine is called
1) after dec(fpas) in SH4 for pluncher documents (as now fpas < sump might hold)
2) in SH4 for printer documents as sump has been increased and now, again, fpas < sump might hold; furthermore the current printer stream may have become non empty.
3) at program exit, where "nfd" is increased
4) by teh printer CM itself, after processing of a form:

```
P(SHS);
if previous form final or number of successive forms of the same program more
   then, saym 20 then privar:= 2 else privar:= 1;
printer selection; V(SHS); P(prisem)
```

After completion of this last P-operation there will be a next form in the now currect stream.

SH4 for the printer forms will have the form:

```
P(SHS);
if fio < maxpor then
   begin pmshr:= 2; request:= current form size;
         V(SHS); P(pmsem); P(SHS) end
                else
   DEC(fio, current form size);
"now the p-o transition can be effectuated, followed by"
INC(fip, current form size);
INC(sump, current form size);
printer selection;
V(SHS)
```

Remark. In the procedure "use fpas for waking" a "dec(fpas) may occur, but nevertheless, there is no need to call the "printer selection" there. For this routine "use fpas for waking" has only an effect when "exist(pmshr = 1), that is, while at some earlier stage, fpas was = 0. The final value of fpas will be = 0 or more: as a result this cannot be responsible for the creation of the situation

"fpas < sump".

Afterthought. The 20 successive forms rule had probably better be overruled if " 0 < nfd of current stream": the program has ended its activity and the danger of monopolization cannot persist very long any more.

Remark on the safety of printer activity control.

A PM may be blocked after the succesful performance of "dec(fpas)" but before succesful performance of "dec(fio)" : the decrease of fpas may be ahaed of that of fio. In that case the printer is sooner urged to activity, the out-of-phaseness can therefore never give rise to output blocking, due to the printer failing to recognize its duty to go to work. On the other hand, it can never be urged to go to work too soon, i.e. without printer forms actually present in the buffer for sump must be positive. The increase of sump takes place in the same critical section as the chainigg on, it can be regarded as "in phase".

Strategy and output limitation.

Only in one case -selection of unfinished pluncher documents- we have specified in which order the documents are scanned: in that of decreasing length. But we have more to specify:
1) if a pluncher makes a choice between finished documents, from which stream will it select?
2) if the printer (with privar = 2) finds streams with positive nfd, from which does it select?
3) if the printer (with privar = 2) has to select a stream on account of "fpas < sump", which one has to be selected?

and, as counter part
4) if in SH6 it is inspected which PM with pmshr = 2 can be woken up, which one is selected?

First we remark, that our targets are twofold: we wish to keep the output equipment as busy as possible and should avoid that in some "active streams" the actual speed is zero, because all available outputarea has been usurped by another stream. This can be remedied by keeping the streams as much as possible, equally long; P.A.Voorhoeve has suggested to achieve this via favouring in the removal of pmshr = 2 (building up of the stream) the shortest one and by selection by the CM's favouring the longest ones. This seems to have very attractive properties, if all PM are filled with output producing programs. However: if a PM has processed a program that produced a little bit of output in an empty stream, and no next program produces further output in this stream, the stream length may be so short, that it is never selected for consumption. To remedy this, we could try to introduce a kind of "ageing factor".

We introduce a universal "portion clock" which is increased by a fixed amount "alpha" at each completed SH4; furthermore we note for each stream the value of the portion clock at the latest SH4 completion with respect to this stream; call this stream variable "attachment time". Furthermore, we keep track of the stream length, measured in portions.

If the CM has the freedom to select finished documents or forms, it selects the stream with the maximum value of

"length in portions - attachment time"

If a production at a stream stops, so that the stream may be very short, the CM's will in due time select this stream: the stream will get empty as desired.

When we have to select which PM pmshr = 2 has to be removed, we take the stream with minimal

"length in portions + attachment time"

thus giving some preference to the stream to which the previous completed SH4 was longest ago. The behaviour with , say "alpha = 1/4" seems, up till now, very reasonable.

Speed regulation of the input CM's.

At the moment of i-increase, all four quatities fi, fip, fio and fiop have to be decreased. The idea is that
a) the input CM tries ti perform "dec(fi)" and "dec(fip)" immediately; if unsuccesful, there will be a disaster signalling.
b) then the desirability of "dec(fio)" and "dec(fiop)" is investigated. If these latter decreases are undesirable, the input CM will be retarded on this criterion.

We must bear in mind, that "disaster" may be unavoidable, because the readers are kept outside the banker's algorithm. We have put two requirements on our stategy
1) in granting "dec(fio) and "dec(fiop)" the system sould be somewhat conservative, in order to avoid tapes running in quite unnecessarily and thus provoking a quite unnecessary "fi" or "fip"-disaster. (Or even: hindering output buffering much more than defendable.)
2) in case of effective distaer, however, the speed regulation of the input CM's must be such, that the fi- and fip-alarm's must get the opportunity to signal it. (If one slows down the speed of the input CM's so that fi- and fip-alarm is too carefully avoided, one has to build in another mechanism to detect, that the complete system has got stuck.) WE have chosen the following rules:
1) If no request for a free tape reader is pending, an input CM, wishing to perform "dec(fio)" and "dec(fiop)" will get permission to do so if

"2 * current length of information stream $<$ min(fio, fiop)"

2) if at least one request for a free tape reader is pending, an input CM, wishing to decrease fio and fiop will get permission to do so if

"0 $<$ min(fio, fiop)".

The points at which the need for input CM reactivation must be inspected are
1) at the end of the i-p-transition (where the current length of the information stream has been decreased).
2) at SHo (o-decrease), where both inc(fio) and inc(fiop) have been performed.
3) at the moment where free tape reader request has been refused.

Remark 1. The first cause of removal will never get an opportunity if the stream is empty. Then, however, it will only stop if fio = 0 or fiop = 0, conditions which, after succesful dec(fi) and Dec(fip), imply active output.

Remark 2. At SHo the output CM will first inspect, whether input CM's that require decrease of both fio and fiop- should be reactivated; after that the fiop is used to help PM's waiting at SH1 and the fio is used to help PM's waiting at SH4.

Remark 3. If an input CM builds up a new portion, it might simultaneously request a portion in transput area:

```
P(SHS);
if fi = 0 or fip = 0 then  goto disaster;
dec(fi); dec(fip);
if (if free tape reader requested then 0 else 2 * length of current stream)
                < min(fio,fiop) then
begin dec(fio); dec(fiop); V(icmsem) end
                                  else
begin icmvar:= "characteristic value" end;
V(SHS);

build up next portion

P(icmsem)

now the i-increase can be effected.
```

Every process, granting a segment to an input CM must perform (within a section critical to SHS)

"icmvar:= 0; dec(fio); dec(fiop); V(icmsem)" ;

this may take place during the whole period of building up the next portion. (this technique of a pending request, already pending during the activity of the input CM is a noverlt; it is however perfectly sound.)

Remark 4. If a request for a free tape reader is pending, both readers will read at maximum speed. This is somewhat naieve, but we could not convince ourselves that more elaborate constructions were significantly better.

The description of document selection for input CM's, although already designed, will be postponed, being closely related to the structure of the keyboard program.