

Interruption de processus

☞ Interruption

INTERRUPT [THE [ABOVE]] Processus [CALLED] Variable-Pointeur

☞ Reprise

RESUME [THE [ABOVE]] Processus [CALLED] Variable-Pointeur



Preamble

Preamble

Last Column is 132

Normally Mode is undefined...

Permanent Entities.....

Every Classe has a Moyenne.Service,
a Taux.Moyen,
a Inter. Arrivee,
a Temps.Service,
a Temps.Attente

Define Moyenne.Service,
Taux.Moyen,
Inter. Arrivee as Real Variables

Define Temps.Service,
Temps.Attente as Double Variables



Preamble

```
Every Cpu has      a Numero,
                  a Tache.En.Execution,
                  an Etat

Define Numero, Tache.En.Execution, Etat  as an Integer Variable

The System owns a FileAttente

Processes.....
Every Generateur.Arrivee has a No.Classe

Define No.Classe      as an integer variable

Every Tache has      a Tk.Classe,
                    a Tk.Temps.Execution,
                    a Tk.Priorite
                    and may belong to a FileAttente

Define Tk.Classe, Tk.Priorite  as integer variable
Define Tk.Temps.Execution    as double variable
```

Preamble

```
Define FileAttente as a set ranked by low Tk.Temps.Execution
and then by high Tk.Priorite

Event.....
include Fin.Simulation

" Déclaration des fonctions

Define Choix.F as an Integer Function with 0 arguments

" Declaration des variables statistiques

Tally moy.Attente as the mean,
min.Attente as the minimum,
max.Attente as the maximum of Temps.Attente

Tally moy.Temps.Execution as the mean of Temps.Service

Accumulate Pour.Utilisation as the mean of Etat

Define Forever to mean While ( 1 = 1 )

and " Preamble
```

Tache

```

Process Tache ( No.Classe, Priorite )
  Define No.Classe, Priorite as integer variable
  Define Temps.Arrivee as a real variable
  Define Tache.En.Attente as an integer variable
  Define Le.Cpu as an integer variable
  Define .Aucune to mean 0
  Define .Libre to mean 0
  Define .Occupe to mean 1

  Let Tk.Temps.Execution(Tache) = Exponential.F ( Moyenne.Service (No.Classe), No.Classe)

  for each cpu with u.Cpu(Cpu) > 0
    find the first case
    if found,
      Le.Cpu = Cpu
    else
      Le.Cpu = Choix.F
      File Tache in FileAttente
      Suspend
    endif
  
```



Tache

```

Temps.Arrivee = Time.V

Temps.Attente(No.Classe) = Time.V - Temps.Arrivee
Temps.Arrivee = Time.V

Let Etat(Le.Cpu) = .Occupe
Let Tache.En.Execution ( Le.Cpu) = Tache
Work Tk.Temps.Execution(Tache) Minutes
Temps.Service ( No.Classe) = Time.V - Temps.Arrivee
Let Etat(Le.Cpu) = .Libre

if FileAttente is not empty,
  Remove first Tache.En.Attente from FileAttente
  Reactivate the Tache called Tache.En.Attente now
endif

Let Tache.En.Execution ( Le.Cpu) = .Aucune

end " Tache
  
```



CPU avec interruption

Preamble

Last Column is 132

Normally Mode is undefined...

Permanent Entities

Every Classe has a Moyenne.Service,
a Taux.Moyen,
a Inter.Arrivee,
a Temps.Service,
a Temps.Attente

Define Moyenne.Service,
Taux.Moyen,
Inter.Arrivee as Real Variables
Define Temps.Service,
Temps.Attente as Double Variables

Every Cpu has a Numero,
a Tache.En.Execution,
an Etat

Define Numero, Tache.En.Execution, Etat as an Integer Variable



CPU avec interruption

The System owns a FileAttente,
and owns a TachesInterromptues

Processes

Every Generateur.Arrivee has a No.Classe

Define No.Classe as an integer variable

Every Tache has a Tk.Classe,
a Tk.Temps.Execution,
a Tk.Priorite
and may belong to a FileAttente,
and may belong to a TachesInterromptues

Define Tk.Classe, Tk.Priorite as integer variable
Define Tk.Temps.Execution as double variable

Define FileAttente as a set ranked by low Tk.Temps.Execution
and then by high Tk.Priorite

Define TachesInterromptues as a set ranked by low Tk.Temps.Execution
and then by high Tk.Priorite



CPU avec interruption

```
Event.....  
    include Fin.Simulation  
  
" Déclaration des fonctions  
  
Define Choix.F as an Integer Function with 0 arguments  
  
" Declaration des variables statistiques  
  
Tally moy.Attente as the mean,  
    min.Attente as the minimum,  
    max.Attente as the maximum of Temps.Attente  
  
Tally moy.Temps.Execution as the mean of Temps.Service  
  
Accumulate Pour.Utilisation as the mean of Etat  
  
Define Forever to mean While ( 1 = 1 )  
  
end " Preamble
```



CPU avec interruption

```
Main  
  
Call Initialiser  
  
for each Classe,  
do  
    Activate a Generateur.Arrivee ( Classe ) now  
loop  
  
Schedule a Fin.Simulation in 4 * 720 Minutes  
  
Start Simulation  
  
end " Main
```



CPU avec interruption

```
Process Generateur.Arrivee ( No.Classe )
  Define No.Classe as an integer variable

  Forever
  Do
    Wait Exponential.F ( Inter.Arrivee(No.Classe), 5 ) Minutes
    Activate a Tache ( No.Classe, N.Classe - No.Classe ) Now
  Loop

end "Generateur.Arrivee
```



CPU avec interruption

```
Routine Initialiser

print 1 line thus
Donnez le nombre de classes

Read N.Classe
Create All Classe
For Each Classe,
Do
  Print 1 line with Classe thus
  Classe *** Donnez la Moyenne et Taux moyen
  Read Moyenne.Service(Classe), Taux.Moyen(Classe)
  Inter.Arrivee (Classe ) = 1.0 / Taux.Moyen(Classe)
Loop

print 1 line thus
Donnez le nombre de CPU
Read N.Cpu

Create all Cpu
for each Cpu,
do
  let Numero ( Cpu ) = Cpu
loop
end " Initialiser
```



CPU avec interruption

```

Process Tache ( No.Classe, Priorite )
  Define No.Classe, Priorite as integer variable
  Define Temps.Arrivee as a real variable

  Define Tache.A.Interrompre as an integer variable
  Define Le.Cpu as an integer variable

  Define .Aucune to mean 0
  Define .Libre to mean 0
  Define .Occupe to mean 1

  Let Tk.Temps.Execution(Tache) = Exponential.F ( Moyenne.Service (No.Classe), No.Classe)
  Temps.Arrivee = Time.V

  for each cpu with Etat(Cpu) = .Libre
    find the first case
  if found,
    Le.Cpu = Cpu
  else
    for each Cpu with Priorite > Tk.Priorite (Tache.En.Execution(Cpu))
      and Tk.Temps.Execution(Tache) < Tk.Temps.Execution(Tache.En.Execution(Cpu)),
      find the first case
  
```



CPU avec interruption

```

if found,
  Le.Cpu = Cpu
  Tache.A.Interrompre = Tache.En.Execution(Le.Cpu)
  Interrupt The Tache Called Tache.A.Interrompre
  File Tache.A.Interrompre in TachesInterromptues
else
  Le.Cpu = Choix.F
  File Tache in FileAttente
  Suspend
endif
endif

Temps.Attente(No.Classe) = Time.V - Temps.Arrivee
Temps.Arrivee = Time.V

Let Etat(Le.Cpu) = .Occupe
Let Tache.En.Execution ( Le.Cpu) = Tache
Work Tk.Temps.Execution(Tache) Minutes
Temps.Service ( No.Classe ) = Time.V - Temps.Arrivee
Let Etat(Le.Cpu) = .Libre

Call Demarrer.Une.Tache

Let Tache.En.Execution ( Le.Cpu) = .Aucune

end " Tache
  
```



CPU avec interruption

```
Routine Demarrer.Une.Tache
  Define Tache.En.Attente, Tache.Interruptue as an integer variable
  Define Priorite.1, Priorite.2 as integer variable
  Define T.Execution.1, T.Execution.2 as double variable

  T.Execution.1 = RINF.C
  T.Execution.2 = RINF.C

  if FileAttente is not empty,
    Priorite.1 = Tk.Priorite(F.FileAttente)
    T.Execution.1 = Tk.Temps.Execution(F.FileAttente)
  endif

  if TachesInterruptues is not empty,
    Priorite.2 = Tk.Priorite(F.TachesInterruptues)
    T.Execution.2 = Tk.Temps.Execution(F.TachesInterruptues)
  endif

  if FileAttente is not empty OR
     TachesInterruptues is not empty,
    if T.Execution.1 < T.Execution.2,
      Remove first Tache.En.Attente from FileAttente
      Reactivate the Tache called Tache.En.Attente now
    else
      Remove first Tache.Interruptue from TachesInterruptues
      Resume the tache called Tache.Interruptue
    endif
  endif
end "Demarrer.Une.Tache"
```

