

An Operating System Scheduler

Scheduler

The scheduler is the component of an operating system that determines which process should be run, and when.

We will specify:

- the service provided—the scheduler specification
- a system that provides this service—the scheduler implementation

Process states

- Since there is a single processor, at any time, there will be at most one process running. We will call this the current process.
- There may be several processes that are waiting to use the processor. These processes are said to be ready.
- There may be some processes that are waiting, not for the processor, but for a different resource or event. These processes are said to be blocked.

Processes

- there is a single processor to be shared
- this is made available to one process at a time
- a process that is currently making use of the processor is said to be running

Specification

Our system will deal with up to n processes, where n is a natural number.

— $n : \mathbb{N}$

Each process will be associated with a process identifier, or *pid*.

$Pid == 1..n$

Zero is used to represent the 'null process': a marker that says that there is no process where this value is found.

$nullPid == 0$

An 'optional pid' can be either a true pid or the null pid:

$OptPid == Pid \cup \{nullPid\}$

Abstract state

```

AScheduler
  current : P PId
  ready : P PId
  blocked : P PId
  free : P PId
  {current} \ {nullPid},
  ready,
  blocked,
  free) partition PId

```

Initialisation

```

ASchedulerInit
  AScheduler'
  current' = nullPid
  ready' =  $\emptyset$ 
  blocked' =  $\emptyset$ 
  free' = PId

```

Operations

- create a process, adding it to the set of ready processes
- dispatch one of the ready processes to the processor
- timeout a process, removing it from the processor and returning it to the set of ready processes
- block a process, removing it from the processor and adding it to the set of blocked processes
- wake up a blocked process, moving it into the set of ready processes
- destroy a process

Create

```

ACreate
  AScheduler
  pi : PId
  free  $\neq \emptyset$ 
  current' = current
  ready' = ready  $\cup \{pi\}$ 
  blocked' = blocked
  free' = free \ {pi}
  pi  $\in$  free

```

Dispatch

```

ADispatch
  AScheduler
  pi : PId
  current = nullPid
  ready  $\neq \emptyset$ 
  current'  $\in$  ready
  ready' = ready \ {current'}
  blocked' = blocked
  free' = free
  pi = current'

```

Exercises

- write schemas to describe the effects of timeout, block, and wake up.
- write three partial operation schemas to describe the effect of destroying:
 - the current process
 - a ready process
 - a blocked process

Design

- our program will use a simple data structure: an array and a few counters
- our use of this data structure can be modelled using chains: finite injections from PId to PId with a unique start and a unique end.

Chains

$\begin{aligned} \text{Chain} & \text{---} \\ \text{start, end} & : \text{OptPId} \\ \text{links} & : \text{PId} \rightsquigarrow \text{PId} \\ \text{set} & : \mathbb{F} \text{PId} \\ \text{set} & = \text{dom links} \cup \text{ran links} \cup \{\text{start}\} \setminus \{\text{nullPId}\} \\ \text{links} & = \emptyset \Rightarrow \text{start} = \text{end} \\ \text{links} & \neq \emptyset \Rightarrow \\ & \{ \text{start} \} = (\text{dom links}) \setminus \text{ran links} \\ & \{ \text{end} \} = (\text{ran links}) \setminus \text{dom links} \\ \forall e : \text{set} \mid e \neq \text{start} \bullet \text{start} \rightarrow e \in \text{links}^+ \end{aligned}$

Exercises

- how do we know that the *start* and the *end* of a given chain are uniquely defined?
- what must be true of the *start* and the *end* pids if set is empty?

Initialisation

$\begin{aligned} \text{ChainInit} & \text{---} \\ \text{Chain}' & \text{---} \\ \text{start}' & = \text{nullPId} \text{end}' = \text{nullPId} \end{aligned}$
--

Operations

- push an element onto the end of a chain
- pop an element from the front of a chain
- delete an element from a chain

Pop

$\begin{aligned} \text{PopSingleton} & \text{---} \\ \Delta \text{Chain} & \text{---} \\ \text{pi} : \text{PId} & \text{---} \\ \text{start} \neq \text{nullPId} & \text{---} \\ \text{links} = \emptyset & \text{---} \\ \text{start}' = \text{nullPId} & \text{---} \\ \text{links}' = \text{links} & \text{---} \\ \text{pi}' = \text{start} & \text{---} \end{aligned}$

PopMultiple

Δ Chain

$pl : PId$

links $\neq \emptyset$

start' = links start

links' = {start} \triangleleft links

$pl = start$

Pop $\hat{=}$ PopSingleton \vee PopMultiple

Delete

Delete $\hat{=}$ DeleteStart \vee DeleteMiddle \vee DeleteEnd

DeleteStart

Δ Chain

$p? : PId$

$p? = start$

$\exists pl : PId \bullet Pop$

DeleteEnd

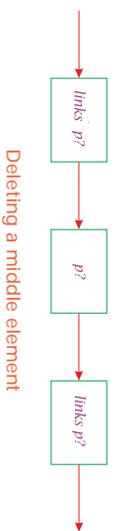
Δ Chain

$p? : PId$

$p? \neq start$

$p? = end$

links' = links \triangleright {end}



DeleteMiddle

Δ Chain

$p? : PId$

$p? \neq start$

$p? \neq end$

links' = {p?} \triangleleft links \oplus {links' $p?$ \rightarrow links $p?$ }

Design

CScheduler

ReadyChain

BlockedChain

FreeChain

current : OptPId

chainstore : PId \rightarrow OptPId

{current} \ {nullPId}, rset, bset, fset) partition PId

rlinks = rset \triangleleft chainstore \triangleright rset

blinks = bset \triangleleft chainstore \triangleright bset

flinks = fset \triangleleft chainstore \triangleright fset

current \neq nullPId \Rightarrow chainstore.current = nullPId

```

ReadyChain ≡
Chain[rstart / start, rend / end, rlinks / links, rset / set]
BlockedChain ≡
Chain[bstart / start, bend / end, blinks / links, bset / set]
FreeChain ≡
Chain[fstart / start, fend / end, flinks / links, fset / set]

```

Initialisation

```

CSchedulerInit
CScheduler'
ReadyChainInit
BlockedChainInit
FreeChainFull
current' = nullPid

```

```

ReadyChainInit ≡
ChainInit[rstart / start, rend / end,
rlinks' / links', rset' / set']
BlockedChainInit ≡
ChainInit[bstart / start, bend / end,
blinks' / links', bset' / set']

```

Operations

```

PushReadyChain ≡
Push[rstart / start, rend / end, rlinks / links, rset / set,
rstart' / start', rend' / end', rlinks' / links', rset' / set']
PopReadyChain ≡
Pop[rstart / start, rend / end, rlinks / links, rset / set,
rstart' / start', rend' / end', rlinks' / links', rset' / set']
PopFreeChain ≡
Pop[fstart / start, fend / end, flinks / links, fset / set,
fstart' / start', fend' / end', flinks' / links', fset' / set']

```

```

CDispatch
ΔCScheduler
pi : Pid
EBlockedChain
EFreeChain
current = nullPid
rset ≠ ∅
PopReadyChain
current' = pi

```

```

CCreate
ΔCScheduler
pi : Pid
EBlockedChain
fset ≠ ∅
current' = current
PopFreeChain
PushReadyChain[pi / p?]

```

Abstract state

```

current = 3
ready = {2, 4, 6}
blocked = {5, 7}
free = {1, 8, 9, 10}

```

Possible concrete state

```

current = 3
chairsStore = {1 ↦ 8, 2 ↦ 6, 3 ↦ 0, 4 ↦ 2, 5 ↦ 0,
               6 ↦ 0, 7 ↦ 5, 8 ↦ 9, 9 ↦ 10, 10 ↦ 0}
rstart = 4
rend = 6
rlinks = {4 ↦ 2, 2 ↦ 6}
rset = {2, 4, 6}

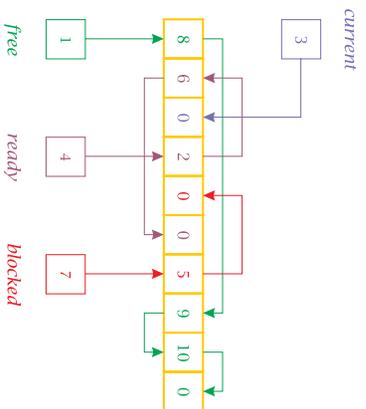
```

```

bstart = 7
bend = 7
blinks = {7 ↦ 5}
bset = {5, 7}

fstart = 1
fend = 10
flinks = {1 ↦ 8, 8 ↦ 9, 9 ↦ 10}
fset = {1, 8, 9, 10}

```

**Retrieve function**

```

RetrScheduler
AScheduler
CScheduler
ready = rset
blocked = bset
free = fset

```

Correctness

$$\begin{aligned}
&CScheduler \vdash E_1 AScheduler \bullet RetrScheduler \\
&CSchedulerInit \wedge Retr' \vdash ASchedulerInit \\
&pre AOp \wedge Retr \wedge COP \wedge Retr' \vdash AOp
\end{aligned}$$