

Laboratorio di Sistemi Operativi  
Anno Accademico 2005-2006

### Kaya Phase 1 Project Specifications

Mauro Morsiani

Copyright © 2006 Senso Davoli, Alberto Montresor, Mauro Morsiani  
Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license can be found at:  
<http://www.gnu.org/licenses/fdl.html>

## Kaya project specifications

### ? **Kaya is:**

- ? a complete Operating System project specification based on Dijkstra's T.H.E. System and Babaoglu and Schneider's HOCA OS
- ? designed as a *multilayered* system, to be developed in a number of *phases*; can be implemented using the uMPS simulator
- ? Phase 1 is the first software layer of Kaya (below it there are only bare hardware and ROM microcode)
- ? Kaya Phase 1 requires the development of a *Queue Manager*

© 2006 Mauro Morsiani

2

## Kaya project specifications

### ? **Why to develop a queue manager?**

- ? because processes and process queues are the basic data structures managed by any kernel
- ? to provide the required ADTs (*Abstract Data Types*) to the upper levels of Kaya
- ? to gain experience in software development and debugging by using the uMPS simulator

© 2006 Mauro Morsiani

3

## Kaya project specifications

### ? **Process Control Block definition (ProcBlk):**

```
/* process control block type */
typedef struct pcb_t {
    /* process queue fields */
    struct pcb_t *p_next, /* pointer to next entry */

    /* process tree fields */
    *p_pnt, /* pointer to parent */
    *p_child, /* pointer to 1st child */
    *p_sib; /* pointer to sibling */

    state_t p_s; /* processor state */
    int *p_semAdd; /* pointer to sema4 on */
                /* which process blocked */

    /* plus other entries to be added later */
} pcb_t;
```

© 2006 Mauro Morsiani

4

## Kaya project specifications

### ? Queue manager goals:

- ? Queue manager = a collection of *modules* that will be used in Phase 2
- ? Should implement the following features:
  - ? allocation and de-allocation of single ProcBlk descriptors
  - ? maintenance of ProcBlk queues
  - ? maintenance of ProcBlk trees
  - ? maintenance of ASL (*Active Semaphore List*): a single sorted list of active semaphore descriptors, each one supporting a ProcBlk queue

## Kaya project specifications

### ? Queue manager features

- ? Allocation and de-allocation of single ProcBlk descriptors:
  - ? no OS = no heap = no dynamic memory allocation
  - ? how to allocate ProcBlk descriptors?
  - ? *static* allocation: define an array of MAXPROC descriptors and use a *pcbFree* list (that is, a list of the unused descriptors), pointed by a *pcbFree\_h* pointer
  - ? MAXPROC will be defined (also) in p1test.c

## Kaya project specifications

### ? Allocation-related functions:

- ? `initPcbs()`  
/\* Initialize the *pcbFree* list to contain all the elements of the array  
`static pcb_t pcbTable[MAXPROC]`  
this method will be called only once during data structure initialization  
\*/
- ? `void freePcb(pcb_t *p)`  
/\* Insert the element pointed to by *p* onto the *pcbFree* list \*/

## Kaya project specifications

### ? Allocation-related functions (cont'd):

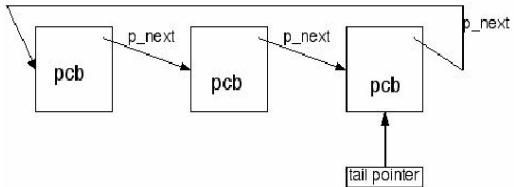
- ? `pcb_t *allocPcb()`  
/\* Return NULL if the *pcbFree* list is empty. Otherwise, remove an element from the *pcbFree* list, provide initial values for ALL of the ProcBlk's fields (i.e. NULL and/or 0) and then return a pointer to the removed element; ProcBlk's get reused, so it is important that no previous value persist in a ProcBlk when it gets reallocated \*/

## uMPS software development

### Queue manager features

#### Maintenance of ProcBlk queues:

- circular, single-linked, tail pointed; using `p_next`
- implementation efficiency may be improved introducing double-linked queues ( and `p_prev`)



© 2006 Mauro Morsiani

9

## Kaya project specifications

### ? Maintenance of ProcBlk queues:

```
? pcb_t *mkEmptyProcQ()
```

```
/* This method is used to initialize a variable to be tail pointer to a process queue; returns a pointer to the tail of an empty process queue, i.e. NULL */
```

```
? int emptyProcQ(pcb_t *tp)
```

```
/* Return TRUE if the queue whose tail is pointed to by tp is empty; return FALSE otherwise */
```

```
? insertProcQ(pcb_t **tp, pcb_t *p)
```

```
/* Insert the ProcBlk pointed to by p into the process queue whose tail-pointer is pointed to by tp; note the double indirection through tp to allow for the possible updating of the tail pointer as well */
```

© 2006 Mauro Morsiani

10

## Kaya project specifications

### ? Maintenance of ProcBlk queues (cont'd):

```
? pcb_t *removeProcQ(pcb_t **tp)
```

```
/* Remove the first (i.e. head) element from the process queue whose tail-pointer is pointed to by tp. Return NULL if the process queue was initially empty; otherwise return the pointer to the removed element. Update the process queue's tail pointer if necessary */
```

```
? pcb_t *outProcQ(pcb_t **tp, pcb_t *p)
```

```
/* Remove the ProcBlk pointed to by p from the process queue whose tail-pointer is pointed to by tp. Update the process queue's tail pointer if necessary. If the desired entry is not in the indicated queue (an error condition), return NULL; otherwise, return p. Note that p can point to any element of the process queue */
```

© 2006 Mauro Morsiani

11

## Kaya project specifications

### ? Maintenance of ProcBlk queues (cont'd):

```
? pcb_t *headProcQ(pcb_t *tp)
```

```
/* Return a pointer to the first ProcBlk from the process queue whose tail is pointed to by tp. Do not remove this ProcBlk from the process queue. Return NULL if the process queue is empty */
```

© 2006 Mauro Morsiani

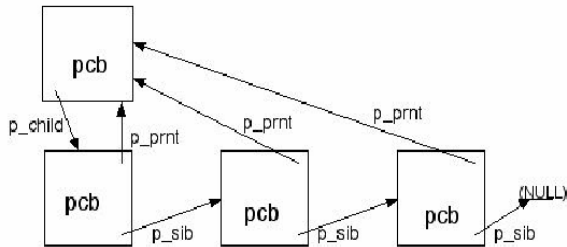
12

## uMPS software development

### Queue manager features

#### Maintenance of ProcBlk trees:

- implemented using `p_prnt`, `p_child`, `p_sib`



## Kaya project specifications

### ? Maintenance of ProcBlk trees:

```
? int emptyChild(pcb_t *p)
```

```
/* Return TRUE if the ProcBlk pointed to by p has no children. Return FALSE otherwise. */
```

```
? insertChild(pcb_t *prnt, pcb_t *p)
```

```
/* Make the ProcBlk pointed to by p a child of the ProcBlk pointed to by prnt */
```

```
/* in other words: insert p in the process tree as a child of prnt */
```

## Kaya project specifications

### ? Maintenance of ProcBlk trees (cont'd):

```
? pcb_t *removeChild(pcb_t *p)
```

```
/* Make the first child of the ProcBlk pointed to by p no longer a child of p. Return NULL if initially there were no children of p. Otherwise, return a pointer to this removed first child ProcBlk. */
```

```
/* in other words: if p has children, remove the first one from the tree and return a pointer to it; if p has no children, return NULL */
```

```
? pcb_t *outChild(pcb_t *p)
```

```
/* Make the ProcBlk pointed to by p no longer the child of its parent. If the ProcBlk pointed to by p has no parent, return NULL; otherwise, return p. Note that the element pointed to by p need not be the first child of its parent. */
```

```
/* in other words: look in the tree for the ProcBlk pointed to by p: if it has no parent, return NULL; if it does have a parent, remove p from the tree and return p */
```

## uMPS software development

### Queue manager features

ASL (*Active Semaphore List*):

- A semaphore is an integer
- A semaphore is *active* if at least one ProcBlk is associated to it

Suggested ASL implementation:

- sorted NULL-terminated single-linked list pointed to by `sem_d_h`, built using `s_next` pointer
- `s_procQ` is the tail pointer of a ProcBlk queue
- a free list of `MAXPROC` elements is required (as for ProcBlks)

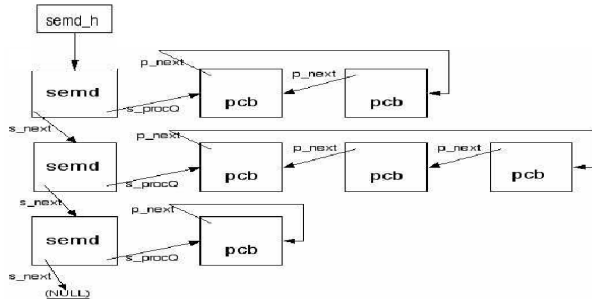
```
/* semaphore descriptor type */
```

```
typedef struct semdt {
    struct semdt *s_next; /* next element on the ASL */
    int *s_semAdd; /* pointer to the semaphore */
    pcb_t *s_procQ; /* tail pointer to a */
    /* process queue */
} semdt;
```

## uMPS software development

### ASL Maintenance

ASL structure:



© 2006 Mauro Morsiani

17

## Kaya project specifications

### ? ASL maintenance:

```
? initASL()
```

```
/* Initialize the semdFree list to contain all the elements of the array  
static semd_t semdTable[MAXPROC]
```

```
This method will be only called once during data structure initialization  
*/
```

```
? pcb_t *headBlocked(int *semAdd)
```

```
/* Return a pointer to the ProcBlk that is at the head of the process  
queue associated with the semaphore semAdd. Return NULL if  
semAdd is not found on the ASL or if the process queue associated  
with semAdd is empty */
```

© 2006 Mauro Morsiani

18

## Kaya project specifications

### ? ASL maintenance (cont'd):

```
? int insertBlocked(int *semAdd, pcb_t *p)
```

```
/* Insert the ProcBlk pointed to by p at the tail of the process queue  
associated with the semaphore whose physical address is semAdd  
and set the semaphore address of p to semAdd. If the semaphore  
is currently not active (i.e. there is no descriptor for it in the ASL),  
allocate a new descriptor from the semdFree list, insert it in the  
ASL (at the appropriate position), initialize all of the fields (i.e. set  
s_semAdd to semAdd, and s_procq to mkEmptyProcQ()), and  
proceed as above. If a new semaphore descriptor needs to be  
allocated and the semdFree list is empty, return TRUE. In all other  
cases return FALSE */
```

© 2006 Mauro Morsiani

19

## Kaya project specifications

### ? ASL maintenance (cont'd):

```
? pcb_t *removeBlocked(int *semAdd)
```

```
/* Search the ASL for a descriptor of this semaphore. If none is found,  
return NULL; otherwise, remove the first (i.e. head) ProcBlk from the  
process queue of the found semaphore descriptor and return a pointer to  
it. If the process queue for this semaphore becomes empty after this  
(emptyProcQ(s_procq) is TRUE), remove the semaphore descriptor  
from the ASL and return it to the semdFree list */
```

```
? pcb_t *outBlocked(pcb_t *p)
```

```
/* Remove the ProcBlk pointed to by p from the process queue associated  
with p's semaphore (p->p_semAdd) on the ASL. If ProcBlk pointed to by  
p does not appear in the process queue associated with p's semaphore,  
which is an error condition, return NULL; otherwise, return p */
```

© 2006 Mauro Morsiani

20

# Kaya project specifications

---

## ? **Some observations on Phase 1:**

- ? Specifications may not look clear:
  - ? you have to think there is a bigger picture
  - ? focus on module structure and features
- ? Writing the code:
  - ? start writing stubs; build one step at a time
  - ? keep it simple
  - ? check for error conditions (plan for the unforeseen)
  - ? analyze and understand `pltest.c`
  - ? look at the examples provided with uMPS
- ? remember `#define HIDDEN static` for module data structures