pandOS

The PandOS operating system is an educational project consisting in the implementation of a kernel/OS designed to run on μMPS. This documentation describes the implementation for the first phase of the project.

DESIGN CHOICES

Platform

The pandOS repository has been developed on GitHub to help the authors to cooperate on the project, combined with GitFlow to manage branches.

Building

To automate the building process CMake was adopted for the generation of the makefile.

Documentation

The guideline to write the documentation is the Doxygen standard, used to have a consistent way to comment the functions. For the next phases the automated Doxygen documentation generator will be used so the same style is preserved through the next phases.

MODULES

PANDOS_CONST

This header file contains utility constants & macro definitions.
In addition to the pre-existing ones, some more constants have been declared.

MAXPROC
Max number of concurrent processes pandOS can support.

MININT
Identifier with the lowest value, used for the first dummy semaphore at the start of the ASL.

MAXINT
Identifier with the highest value, used for the second dummy semaphore at the end of the ASL.

MAXSEM
Total number of semaphores to be inserted in the ASL, counting also the 2 dummies ones.

PANDOS_TYPES

This header file contains utility types definitions. It defines:

typedef signed int cpu_t

typedef unsigned int memaddr

typedef struct pcb_t

Process Control Blocks (pcbs). It contains the following members:

struct pcb_t	*p_next	ptr to next entry
struct pcb_t	*p_prev	ptr to previous entry
struct pcb_t	*p_prnt,	ptr to parent
struct pcb_t	*p_child,	ptr to 1st child
struct pcb_t	*p_next_sib,	ptr to next sibling
struct pcb_t	*p_prev_sib	ptr to prev. sibling
state_t	p_s	processor state
cpu_t	p_time	cpu time used by proc
int	*p_semAdd	ptr to semaphore on which proc is blocked

typedef struct semd_t

Active Semaphore List (ASL). It contains the following members:

struct semd_t	*s_next	ptr to next element on queue
int	*s_semAdd	ptr to the semaphore

PCB

HIDDEN pcb_t *pcbFree_h NULL-terminated single, linearly linked list containg the unused PCBs

Process Control Blocks functions

HIDDEN pcb_t *resetPcb(pcb_tp)

Resets all the values of a pcb pointer to NULL.

Parameters

p The pointer to the PCB that has to be resetted.

Returns

The pointer to the pcb.

void initPcbs()

Initializes the pcbFree list. This function should be called only once during initialization.

void freePcb(pcb_t *p)

Deallocates the element pointed by p.

Parameters

p Pointer to the pcb that has to be inserted in the pcbFree list.

pcb_t *allocPcb()

Allocates a PCB and initializes his values.

Returns

NULL if the pcbFree list is empty otherwise a pointer to the removed pcb.

void initPcbs()

Initializes the pcbFree list. This function should be called only once during initialization.

pcb_t *mkEmptyProcQ()

Initializes a new empty process queue.

Returns

A tail pointer to an empty process queue.

int emptyProcQ(pcb_t *tp)

Checks if the queue pointed by tp is empty.

Parameters

tp Tail pointer of the queue.

Returns

TRUE if the queue is empty, FALSE otherwise.

void insertProcQ(pcb_t **tp, pcb_t *p)

Inserts the pcb pointed by p into the queue pointed by tp.

Parameters

tp Tail pointer of the queue.

p Pointer to the pcb.

pcb_t *headProcQ(pcb_t *tp)

Returns the pointer to the head of the tp process queue, without removing it.

Parameters

tp The pointer to the tail of the process queue.

Returns

The pointer to the head of the process queue, NULL if the queue is empty.

pcb_t *removeProcQ(pcb_t **tp)

Removes the oldest element (the head) from the tp queue.

Parameters

tp The pointer to the queue.

Returns

The pointer to the element removed from the list, NULL if the queue is empty.

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

Removes the PCB pointed by P from the process queue pointed by tp.

Parameters

tp The pointer to the queue.

p The pointer to the PCB that has to be removed.

Returns

The pointer to the removed PCB, NULL if the PCB pointed by p is not in the queue.

Definitions of Process Tree functions

HIDDEN pcb_t *trim(pcb_t *p)

This funcion takes as input a pointer to a PCB who has to be removed from his tree.

Parameters

p The pcb pointer that has to be removed from his tree

Returns
The pointer to the PCB whose fields have been set to NULL

int emptyChild(pcb_t *p)

Inspects if the PCB pointed by p has a child.
Parameters
p The pointer to the PCB that has to be inspected.
Returns
TRUE if the PCB pointed by p has no children, FALSE otherwise.

void insertChild(pcb_t *prnt, pcb_t *p)

Inserts the PCB pointed by p as a child of the PCB pointed by prnt.
Parameters
prnt The pointer to the PCB which will become parent of p.
p The pointer to the PCB which will become child of prnt.

pcb_t *removeChild(pcb_t *p)

Removes the first child of the PCB pointed by p.
Parameters
p The pointer to the PCB whose first child will be removed.
Returns
The pointer to the first child of the PCB, NULL if the PCB doesn't have a child.

pcb_t *outChild(pcb_t *p)

Removes the PCB pointed by p from the list of his parent's children.
Parameters
p The pointer to the PCB that will be removed.
Returns
The pointer to the PCB, NULL if the PCB doesn't have a parent.

ASL

Active Semaphore List functions. It defines:

HIDDEN semd_t* semdFree_h NULL-terminated single, linearly linked unused semaphore list

HIDDEN semd_t* semd_h NULL-terminated single, linearly linked active semaphore list

HIDDEN semd_t *findPrevSem(int *semAdd)

This function takes as input a semAdd and returns the last semaphore in semd_h whose identifier is lower than the one passed as argument.

Parameters

semAdd Semaphore identifier

Returns

The last semaphore whose semaphore is lower than semAdd.

int insertBlocked(int *semAdd,pcb_t *p)

Insert the pcb 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.

Parameters

semAdd Semaphore identifier.

p Pointer to the PCB to be inserted.

Returns

TRUE if a new semaphore descriptor needs to be allocated, FALSE otherwise.

pcb_t *removeBlocked(int *semAdd)

Search for a semaphore whose descriptor is semADD. Remove the first pcb from its process queue and return apointer to it.

Parameters

semAdd Semaphore identifier.

Returns

The pointer to the head from the process queue associated with the semaphore descriptor.

pcb_t *outBlocked(pcb_t *p)

Remove the pcb pointed to by p from the process queue associated with p's semaphore.

Parameters

p Pointer to the pcb to be removed.

Returns

A pointer to the removed PCB. Returns NULL if p does not appear in the process queue.

pcb_t *headBlocked(int *semAdd)

The a pointer to the head of the process queue associated with the semaphore semAdd

Parameters

semAdd Semaphore identifier.

Returns

The first element of the process queue associated with the semaphore semAdd or NULL if semAdd is not found.

void initASL()

Initialize the semdFree list, this method will be only called once during data structure initialization.