**JAROS**

**Programmer’s Manual**

**Group 3**

Jonroy Canady | Adam Trainer | Robert Wayland

Table of Contents

Overview of COMHAN / MPX 4

Functions 5

int cleanup\_r1(); 5

int comhan(); 5

int date(); 5

int disp\_dir(); 5

int help(); 5

int init\_r1(); 5

int valid\_date(int yr, int mo, int day); 5

void err\_hand(int err\_code); 5

void get\_Version(); 6

void terminate\_mpx(); 6

void toLowerCase(char str[BIGBUFF]); 6

void trim(char ary[BIGBUFF]); 6

int init\_r2(); 6

int cleanup\_r2(); 6

int block(); 6

int unblock(); 6

int suspend(); 7

int resume(); 7

int set\_Priority(); 7

int show\_PCB(); 7

int show\_All(); 7

int show\_Ready(); 7

int show\_Blocked(); 7

struct PCB \* allocate\_PCB(); 8

int setup\_PCB(struct PCB \*PCBptr, char name[PROCESS\_NAME\_LENGTH], int proc\_class, int priority); 8

int free\_PCB(struct PCB \*PCBptr); 8

int create\_PCB(); 8

int delete\_PCB(); 8

int isEmpty(int q); 8

int insert(struct PCB \*newPCB, int q); 8

struct PCB\* findPCB(char \*name, struct PCB \*PCBptr); 8

struct PCB\* qRemove(char \*name, struct PCB \*set); 9

void toLowerCasex(char str[BIGBUFF]); 9

void trimx(char ary[BIGBUFF]); 9

struct PCB\* getRHead(); 9

int init\_r3(); 9

int cleanup\_r3(); 9

void interrupt sys\_call(); 9

void interrupt dispatcher(); 9

int load\_test(); 9

int load\_prog(char fname[], int pri); 10

int terminate(); 10

int load(); 10

int com\_open (int \*eflag\_p, int baud\_rate); 10

int com\_read(char\* buf\_p, int \*count\_p); 10

int com\_write(char \*buf\_p, int \*count\_p); 10

int com\_close(); 10

void interrupt com\_check(); 10

void readCom(); 10

void writeCom(); 11

void stop\_com\_request(); 11

int com\_open (int \*eflag\_p, int baud\_rate) 11

int com\_read(char\* buf\_p, int \*count\_p) 11

int com\_write(char \*buf\_p, int \*count\_p) 11

Data Structures 11

PCB 11

context 12

Global Variables 12

Cross Reference 13

int cleanup\_r1(); 13

int comhan(); 13

int date(); 13

int disp\_dir(); 13

int help(); 13

int init\_r1() 13

int valid\_date(int yr, int mo, int day) 14

void err\_hand(int err\_code) 14

void get\_Version(); 14

void terminate\_mpx(); 14

void toLowerCase(char str[BIGBUFF]) 14

void trim(char ary[BIGBUFF]) 14

int init\_r2() 14

int cleanup\_r2() 14

int block() 14

int unblock() 14

int suspend() 14

int resume() 15

int set\_Priority(); 15

int show\_PCB() 15

int show\_All() 15

int show\_Ready() 15

int show\_Blocked() 15

struct PCB \* allocate\_PCB() 15

int setup\_PCB(struct PCB \*PCBptr, char name[PROCESS\_NAME\_LENGTH], int proc\_class, int priority) 15

int free\_PCB(struct PCB \*PCBptr) 15

int create\_PCB() 15

int delete\_PCB() 15

int isEmpty(int q) 15

int insert(struct PCB \*newPCB, int q) 16

struct PCB\* findPCB(char \*name, struct PCB \*PCBptr) 16

struct PCB\* qRemove(char \*name, struct PCB \*set) 16

void toLowerCasex(char str[BIGBUFF]) 16

void trimx(char ary[BIGBUFF]) 16

struct PCB\* getRHead() 16

int init\_r3() 16

int cleanup\_r3() 16

void interrupt sys\_call() 16

void interrupt dispatcher() 16

int load\_test() 17

int load\_prog(char fname[], int pri) 17

int terminate() 17

int com\_open (int \*eflag\_p, int baud\_rate) 17

int com\_read(char\* buf\_p, int \*count\_p) 17

int com\_write(char \*buf\_p, int \*count\_p) 17

int com\_close() 17

void interrupt com\_check() 17

void readCom() 17

void writeCom() 17

void stop\_com\_request() 17

int com\_open (int \*eflag\_p, int baud\_rate) 17

int com\_read(char\* buf\_p, int \*count\_p) 18

int com\_write(char \*buf\_p, int \*count\_p) 18

Index 18

# Overview of COMHAN / MPX

The MultiProgramming eXecutive, or MPX, is an experimental platform developed to realistically model various aspects of operating system design. Unlike UNIX or Windows, MPX is not intended to be a fully functional operating system. Instead, each module represents an integral part of an operating system such as the user interface, process management, program management, memory management, etc. An interactive command handler, known as COMHAN, interacts with the keyboard and display to facilitate user input and computer-generated output respectively. The necessary components for MPX are as follows:

* a PC using a processor with the Intel 80x86 architecture and the IBM-PC memory and interrupt structure, a VGA or better display, and at least one USB port
* a Microsoft MS-DOS or Windows family operating system
* an ANSI C compiler (Borland Turbo C/C++ is recommended)

# Functions

## int cleanup\_r1();

* Parameters: N/A
* Return Value: int – an error code
* Description: The cleanup\_r1 function frees memory.

## int comhan();

* Parameters: N/A
* Return Value: int – an error code
* Description: The comhan function processes user-inputted commands and receives error codes returned by functions.

## int date();

* Parameters: N/A
* Return Value: int – an error code
* Description: The date function displays the current date and prompts the user as to whether or not he wants to change the system date.

## int disp\_dir();

* Parameters: N/A
* Return Value: int – an error code
* Description: The disp\_dir function opens the current working directory, displays the contents of the directory, and closes the directory.

## int help();

* Parameters: N/A
* Return Value: int – an error code
* Description: The help function displays help information for MPX and for each command.

## int init\_r1();

* Parameters: N/A
* Return Value: int – an error code
* Description: The init\_r1 function initializes the current working directory.

## int valid\_date(int yr, int mo, int day);

* Parameters: int - year, int - month, int - day
* Return Value: int - a boolean value
* Description: The valid\_date function verifies that the user-inputted date is a valid date. It checks for leap years and checks for a valid number of days in a month.

## void err\_hand(int err\_code);

* Parameters: int - an error code
* Return Value: int – an error code
* Description: The err\_hand function handles errors generated during JAROS execution.

## void get\_Version();

* Parameters: N/A
* Return Value: void
* Description: The get\_Version function displays a brief description of the version of JAROS that is currently running.

## void terminate\_mpx();

* Parameters: N/A
* Return Value: void
* Description: The terminate\_mpx function stops execution of JAROS and returns to the host operating system.

## void toLowerCase(char str[BIGBUFF]);

* Parameters: char [] - a buffer containing user input
* Return Value: void
* Description: The toLowerCase function converts user input to lowercase.

## void trim(char ary[BIGBUFF]);

* Parameters: char [] - a buffer containing user input
* Return Value: void
* Description: The trim function loops through a buffer, removing whitespace and newlines from the user input.

## int init\_r2();

* Parameters: N/A
* Return Value: int – an error code
* Description: The init\_r2 function initializes the current working directory.

## int cleanup\_r2();

* Parameters: N/A
* Return Value: int – an error code
* Description: The cleanup\_r2 function releases all allocated PCBs.

## int block();

* Parameters: N/A
* Return Value: int – an error code
* Description: The block function moves the specified PCB from the READY to BLOCKED queue.

## int unblock();

* Parameters: N/A
* Return Value: int – an error code
* Description: The unblock function places a specified process in the READY state, while retaining its SUSPENDED status. The process is removed from the BLOCKED queue, if necessary, and inserted in the READY queue.

## int suspend();

* Parameters: N/A
* Return Value: int – an error code
* Description: The suspend function places a specified process in a SUSPENDED state. The state chosen will be either SUSPENDED-READY or SUSPENDED-BLOCKED, depending on its previous state.

## int resume();

* Parameters: N/A
* Return Value: int – an error code
* Description: The resume function places a specified process in a non-SUSPENDED state. The state chosen will be either READY or BLOCKED, depending on its previous state.

## int set\_Priority();

* Parameters: N/A
* Return Value: int – an error code
* Description: The set\_Priority function changes the priority of a specified process.

## int show\_PCB();

* Parameters: N/A
* Return Value: int – an error code
* Description: The show\_PCB function displays all information contained in a single PCB for a process specified by name.

## int show\_All();

* Parameters: N/A
* Return Value: int – an error code
* Description: The show\_All function displays all information about all PCBs which are currently in use.

## int show\_Ready();

* Parameters: N/A
* Return Value: int – an error code
* Description: The show\_Ready function displays information about all processes which are currently in the READY or the SUSPENDED-READY state.

## int show\_Blocked();

* Parameters: N/A
* Return Value: int – an error code
* Description: The show\_Blocked function displays information about all processes which are currently in the READY or the BLOCKED state.

## struct PCB \* allocate\_PCB();

* Parameters: N/A
* Return Value: struct PCB \* – an allocated PCB
* Description: The allocate\_PCB function allocates an available PCB, marks it as in use, and returns a PCB pointer or identifier.

## int setup\_PCB(struct PCB \*PCBptr, char name[PROCESS\_NAME\_LENGTH], int proc\_class, int priority);

* Parameters: struct PCB \*, char [], int, int
* Return Value: int – an error code
* Description: The setup\_PCB function initializes the content of a PCB, which is assumed to be newly allocated using allocate\_PCB.

## int free\_PCB(struct PCB \*PCBptr);

* Parameters: N/A
* Return Value: int – an error code
* Description: The free\_PCB function releases an allocated PCB.

## int create\_PCB();

* Parameters: N/A
* Return Value: int – an error code
* Description: The create\_PCB function allocates and setups a new PCB. By default, the process is initially in the READY (not SUSPENDED) state.

## int delete\_PCB();

* Parameters: N/A
* Return Value: int – an error code
* Description: The delete\_PCB function deallocates an existing PCB.

## int isEmpty(int q);

* Parameters: N/A
* Return Value: int – an error code
* Description: The isEmpty function checks to see if the queue is empty.

## int insert(struct PCB \*newPCB, int q);

* Parameters: struct PCB \*, int
* Return Value: int – an error code
* Description: The insert function inserts a PCB into a specified queue. The queue is specified by an identifier which could for example, be the address of its descriptor.

## struct PCB\* findPCB(char \*name, struct PCB \*PCBptr);

* Parameters: char \*, struct PCB \*
* Return Value: struct PCB \* - a PCB
* Description: The findPCB function searches the PCBs for a process having a specified name.

## struct PCB\* qRemove(char \*name, struct PCB \*set);

* Parameters: char \*, struct PCB \*
* Return Value: struct PCB \* - a PCB
* Description: The qRemove function removes a PCB from the queue.

## void toLowerCasex(char str[BIGBUFF]);

* Parameters: char []
* Return Value: void
* Description: The toLowerCasex function converts user input to lowercase.

## void trimx(char ary[BIGBUFF]);

* Parameters: char [] - a buffer containing user input
* Return Value: void
* Description: The trimx function loops through a buffer, removing whitespace and newlines from the user input.

## struct PCB\* getRHead();

* Parameters: N/A
* Return Value: struct PCB \* - a PCB
* Description: The getRHead returns the right most head.

## int init\_r3();

* Parameters: N/A
* Return Value: int – an error code
* Description: The init\_r3 function initializes all global variables for R3.

## int cleanup\_r3();

* Parameters: N/A
* Return Value: int – an error code
* Description: The cleanup\_r3 function releases all allocated PCBs.

## void interrupt sys\_call();

* Parameters: N/A
* Return Value: void
* Description: The interrupt sys\_call interprets system call parameters and invokes the dispatcher as necessary.

## void interrupt dispatcher();

* Parameters: N/A
* Return Value: void
* Description: The interrupt dispatcher function identifies the next READY process, if any, and dispatches it by loading its context from its PCB.

## int load\_test();

* Parameters: N/A
* Return Value: int – an error code
* Description: The load\_test function provides test cases for R3.

## int load\_prog(char fname[], int pri);

* Parameters: char [], int
* Return Value: int – an error code
* Description: The load\_prog function allocates and setups a new PCB. In addition, it allocates program memory using the support procedure sys\_alloc\_mem, and loads a program into that memory using the support procedure sys\_load\_program.

## int terminate();

* Parameters: N/A
* Return Value: int – an error code
* Description: The terminate function terminates a process by deallocating its PCB and releasing its allocated program memory.

## int load();

* Parameters: N/A
* Return Value: int – an error code
* Description: The load function creates a new process.

## int com\_open (int \*eflag\_p, int baud\_rate);

* Parameters: char \*, int \*
* Return Value: int – an error code
* Description: The com\_open function initializes the serial port.

## int com\_read(char\* buf\_p, int \*count\_p);

* Parameters: char \*, int \*
* Return Value: int – an error code
* Description: The load function creates a new process.

## int com\_write(char \*buf\_p, int \*count\_p);

* Parameters: char \*, int \*
* Return Value: int – an error code
* Description: The load function creates a new process.

## int com\_close();

* Parameters: N/A
* Return Value: int – an error code
* Description: The load function creates a new process.

## void interrupt com\_check();

* Parameters: N/A
* Return Value: void
* Description: The load function creates a new process.

## void readCom();

* Parameters: N/A
* Return Value: void
* Description: The load function creates a new process.

## void writeCom();

* Parameters: N/A
* Return Value: void
* Description: The load function creates a new process.

## void stop\_com\_request();

* Parameters: N/A
* Return Value: void
* Description: The load function creates a new process.

## int com\_open (int \*eflag\_p, int baud\_rate)

* Parameters: int \*, int
* Return Value: void
* Description: The com\_open function initializes the serial port.

## int com\_read(char\* buf\_p, int \*count\_p)

* Parameters: char \*, int \*
* Return Value: int – an error code
* Description: The com\_read function obtains input characters and loads them into the requestor's buffer.

## int com\_write(char \*buf\_p, int \*count\_p)

* Parameters: N/A
* Return Value: void
* Description: The load function creates a new process.

# Data Structures

## PCB

* Use: represents a Process Control Block
* Attributes:
  + char name[PROCESS\_NAME\_LENGTH]
  + int id
  + int proc\_class
  + int priority
  + int state
  + int suspended
  + unsigned char stack[STACK\_SIZE]
  + unsigned char\* stack\_base
  + unsigned char\* stack\_top
  + int mem\_size
  + unsigned char\* load\_address
  + unsigned char\* execution\_address
  + struct PCB \*prev
  + struct PCB \*next

## context

* Use: stores a process’ context
* Attributes:
  + unsigned int BP
  + unsigned int DI
  + unsigned int SI
  + unsigned int DS
  + unsigned int ES
  + unsigned int DX
  + unsigned int CX
  + unsigned int BX
  + unsigned int AX
  + unsigned int IP
  + unsigned int CS
  + unsigned int FLAGS

# Global Variables

**int err** – error codes

**char \* fcns[]**  - char array of functions

**char wd[]** – char array of working director

**struct PCB \*tail1** – PCB

**struct PCB \*tail2** - PCB

**struct PCB \*head1 -** PCB

**struct PCB \*head2 -** PCB

**int errx -** error codes

**static unsigned short ss\_save**

**static unsigned short sp\_save**

**static unsigned short ss\_save\_temp**

**static unsigned short sp\_save\_temp**

**static unsigned short new\_ss**

**static unsigned short new\_sp**

**static unsigned char sys\_stack[]**

**static struct PCB \*cop**

**static struct PCB \*tempnode**

**struct context \*context\_p**

**struct params \*param\_p**

**int err3** – error codes

**int err4** – error codes

**static struct DCB \*com\_port** – DCB for the serial port

**static void interrupt (\*oldfunc) (void)**

**static char iochar**

**static char mask**

**static int intType**

# Cross Reference

## int cleanup\_r1();

* Calls: N/A
* Called by: terminate\_mpx,

## int comhan();

* Calls: memset, printf, trim, toLowerCase
* Called by:

## int date();

* Calls: sys\_get\_date, printf, sys\_req, trim, toLowerCase, atoi, err\_hand
* Called by:

## int disp\_dir();

* Calls: memset, strcat, sys\_open\_dir, printf, sys\_get\_entry, sys\_close\_dir
* Called by:

## int help();

* Calls: sys\_req, trim, toLowerCase, strlen, strncmp, strcat, fopen, fgets, printf, err\_hand, fclose
* Called by:

## int init\_r1()

* Calls: \_getdcwd
* Called by:

## int valid\_date(int yr, int mo, int day)

* Calls: N/A
* Called by:

## void err\_hand(int err\_code)

* Calls: printf
* Called by: date, help, terminate\_mpx

## void get\_Version();

* Calls: N/A
* Called by: N/A

## void terminate\_mpx();

* Calls: memset, printf, sys\_req, err\_hand, trim, toLowerCase, cleanup\_r1, cleanup\_r2, cleanup\_r3, sys\_exit
* Called by:

## void toLowerCase(char str[BIGBUFF])

* Calls: N/A
* Called by: comhan, date, help, terminate\_mpx

## void trim(char ary[BIGBUFF])

* Calls: isspace
* Called by: comhan, date, help, terminate\_mpx

## int init\_r2()

* Calls: N/A
* Called by:

## int cleanup\_r2()

* Calls: free\_PCB
* Called by: terminate\_mpx

## int block()

* Calls: memset, printf, sys\_req, trimx, toLowerCasex, findPCB, qRemove, insert
* Called by:

## int unblock()

* Calls: memset, printf, sys\_req, trimx, toLowerCasex, findPCB, qRemove, insert

## int suspend()

* Calls: memset, printf, sys\_req, trimx, toLowerCasex, findPCB, qRemove, insert
* Called by:

## int resume()

* Calls: memset, printf, sys\_req, trimx, toLowerCasex, findPCB
* Called by:

## int set\_Priority();

* Calls: memset, printf, sys\_req, trimx, toLowerCasex, findPCB, atoi, qRemove, insert
* Called by:

## int show\_PCB()

* Calls: memset, printf, sys\_req, trimx, toLowerCasex, findPCB
* Called by:

## int show\_All()

* Calls: printf, sys\_req
* Called by:

## int show\_Ready()

* Calls: printf, sys\_req

## int show\_Blocked()

* Calls: printf, sys\_req
* Called by:

## struct PCB \* allocate\_PCB()

* Calls: sys\_alloc\_mem, sizeof, malloc
* Called by: create\_PCB, load\_test, load\_prog

## int setup\_PCB(struct PCB \*PCBptr, char name[PROCESS\_NAME\_LENGTH], int proc\_class, int priority)

* Calls: strncpy
* Called by: load\_test, load\_prog

## int free\_PCB(struct PCB \*PCBptr)

* Calls: sys\_free\_mem
* Called by: cleanup\_r2, interrupt sys\_call, terminate

## int create\_PCB()

* Calls: printf, sys\_req, findPCB, strncpy, trimx, atoi, allocate\_PCB, insert
* Called by:

## int delete\_PCB()

* Calls: memset, printf, sys\_req
* Called by:

## int isEmpty(int q)

* Calls: strncpy
* Called by: insert

## int insert(struct PCB \*newPCB, int q)

* Calls: isEmpty
* Called by: block, unblock, suspend, set\_Priority, create\_PCB, interrupt sys\_call, load\_test, load\_prog

## struct PCB\* findPCB(char \*name, struct PCB \*PCBptr)

* Calls: strncmp
* Called by: block, unblock, suspend, resume, set\_Priority, show\_PCB, create\_PCB, qRemove,

## struct PCB\* qRemove(char \*name, struct PCB \*set)

* Calls: findPCB
* Called by: block, unblock, suspend, set\_Priority, interrupt dispatcher, terminate

## void toLowerCasex(char str[BIGBUFF])

* Calls: strlen, tolower
* Called by: block, unblock, suspend, resume, set\_Priority, show\_PCB

## void trimx(char ary[BIGBUFF])

* Calls: isspace
* Called by: block, unblock, suspend, resume, set\_Priority, show\_PCB, terminate

## struct PCB\* getRHead()

* Calls: N/A
* Called by: interrupt dispatcher

## int init\_r3()

* Calls: sys\_set\_vec
* Called by:

## int cleanup\_r3()

* Calls: N/A
* Called by: terminate\_mpx

## void interrupt sys\_call()

* Calls: sizeof, insert, free\_pcb, interrupt dispatcher
* Called by:

## void interrupt dispatcher()

* Calls: getRHead, qRemove
* Called by: interrupt sys\_call

## int load\_test()

* Calls: allocate\_PCB, setup\_PCB, insert
* Called by

## int load\_prog(char fname[], int pri)

* Calls: sys\_check\_program, allocate\_PCB, setup\_PCB, sys\_alloc\_mem, sys\_load\_program, insert
* Called by:

## int terminate()

* Calls: memset, printf, sys\_req, trimx, qRemove, free\_PCB
* Called by:

## int com\_open (int \*eflag\_p, int baud\_rate)

* Calls: getvect, setvect, outportb, disable, enable
* Called by:

## int com\_read(char\* buf\_p, int \*count\_p)

* Calls: disable, enable
* Called by:

## int com\_write(char \*buf\_p, int \*count\_p)

* Calls: outportb
* Called by:

## int com\_close()

* Calls: disable, enable, outportb, setvect
* Called by:

## void interrupt com\_check()

* Calls: readCom, writeCom, inportb, outportb
* Called by:

## void readCom()

* Calls: inportb,
* Called by: interrupt com\_check

## void writeCom()

* Calls: outportb, inportb
* Called by:

## void stop\_com\_request()

* Calls: inportb, outportb
* Called by:

## int com\_open (int \*eflag\_p, int baud\_rate)

* Calls: getvect, setvect, outportb, disable, enable
* Called by:

## int com\_read(char\* buf\_p, int \*count\_p)

* Calls: disable, enable
* Called by:

## int com\_write(char \*buf\_p, int \*count\_p)

* Calls: outportb
* Called by:

# Index

\_

\_getdcwd, 13

A

allocate\_PCB, 5, 6, 15, 17

atoi, 12, 14, 15

B

block, 4, 14, 16

C

**char \* fcns[]**, 11

char name[PROCESS\_NAME\_LENGTH], 6, 10, 15

**char wd[]**, 11

cleanup\_r2, 4, 13, 14, 15

cleanup\_r3, 7, 13, 17

COMHAN, 2, 12, 13

**context**, 8, 10, 11

create\_PCB, 6, 15, 16

D

date, 2, 3, 12, 13

disable, 17, 18

E

enable, 17, 18

err\_hand, 3, 12, 13

F

fclose, 13

fgets, 13

findPCB, 7, 14, 15, 16

fopen, 13

free\_PCB, 6, 14, 15, 17

G

getRHead, 7, 16, 17

getvect, 17, 18

H

help, 2, 3, 12, 13

I

inportb, 18

insert, 6, 14, 15, 16, 17

**int cleanup\_r3()**, 7, 17

**int com\_close()**, 9, 18

**int com\_open (int \*eflag\_p, int baud\_rate)**, 8, 9, 17, 18

**int com\_read(char\* buf\_p, int \*count\_p)**, 9, 10, 17, 18

**int com\_write(char \*buf\_p, int \*count\_p)**, 9, 10, 18

**int create\_PCB()**, 6, 15

**int delete\_PCB()**, 6, 15

**int err**, 3, 11, 13

**int err3**, 12

**int err4**, 12

**int errx**, 11

int id, 10

**int init\_r1()**, 3, 13

**int init\_r2()**, 4, 14

**int init\_r3()**, 7, 16

**int insert(struct PCB \*newPCB, int q)**, 6, 16

**int load\_prog(char fname[], int pri)**, 8, 17

**int load\_test()**, 8, 17

**int load()**, 8

int mem\_size, 10

int priority, 6, 10, 15

int proc\_class, 6, 10, 15

**int show\_All()**, 5, 15

**int show\_Blocked()**, 5, 15

**int show\_Ready()**, 5, 15

int state, 10

int suspended, 10

**int terminate()**, 8, 17

**int valid\_date(int yr, int mo, int day)**, 3, 13

interrupt com\_check, 9, 18

interrupt dispatcher, 8, 16, 17

interrupt sys\_call, 8, 15, 16, 17

isEmpty, 6, 15, 16

isspace, 13, 16

L

load\_prog, 8, 15, 16, 17

load\_test, 8, 15, 16, 17

M

malloc, 15

memset, 12, 13, 14, 15, 17

MPX, 2, 3

MultiProgramming eXecutive, 2

O

outportb, 17, 18, 19

P

**PCB**, 4, 5, 6, 7, 8, 10, 11, 15, 16

printf, 12, 13, 14, 15, 17

Q

qRemove, 7, 14, 16, 17

R

readCom, 9, 18

resume, 4, 5, 14, 16

S

set\_Priority, 5, 14, 16

setup\_PCB, 6, 15, 17

setvect, 17, 18

show\_PCB, 5, 14, 16

sizeof, 15, 17

**static char iochar**, 12

**static char mask**, 12

**static int intType**, 12

**static struct DCB \*com\_port**, 12

**static struct PCB \*cop**, 11

**static struct PCB \*tempnode**, 11

**static unsigned char sys\_stack[]**, 11

**static unsigned short new\_sp**, 11

**static unsigned short new\_ss**, 11

**static unsigned short sp\_save**, 11

**static unsigned short sp\_save\_temp**, 11

**static unsigned short ss\_save**, 11

**static unsigned short ss\_save\_temp**, 11

**static void interrupt (\*oldfunc) (void)**, 12

strcat, 12

strlen, 12, 16

strncmp, 12, 16

strncpy, 15

**struct context \*context\_p**, 11

**struct params \*param\_p**, 12

**struct PCB \*head2**, 11

**struct PCB \*tail1**, 11

**struct PCB \*tail2**, 11

**struct PCB\* findPCB(char \*name, struct PCB \*PCBptr)**, 7, 16

**struct PCB\* qRemove(char \*name, struct PCB \*set)**, 7, 16

suspend, 4, 14, 16

sys\_alloc\_mem, 8, 15, 17

sys\_close\_dir, 12

sys\_exit, 13

sys\_free\_mem, 15

sys\_get\_entry, 12

sys\_open\_dir, 12

sys\_req, 12, 13, 14, 15, 17

sys\_set\_vec, 16

T

terminate, 3, 8, 12, 13, 14, 15, 16, 17

terminate\_mpx, 3, 12, 13, 14, 17

tolower, 16

toLowerCase, 3, 12, 13

toLowerCasex, 7, 14, 16

trim, 4, 12, 13

trimx, 7, 14, 15, 16, 17

U

unblock, 4, 14, 16

UNIX, 2

unsigned char stack[STACK\_SIZE], 10

unsigned char\* execution\_address, 10

unsigned char\* load\_address, 10

unsigned char\* stack\_base, 10

unsigned char\* stack\_top, 10

unsigned int AX, 11

unsigned int BP, 10

unsigned int BX, 11

unsigned int CS, 11

unsigned int CX, 11

unsigned int DI, 11

unsigned int DS, 11

unsigned int DX, 11

unsigned int ES, 11

unsigned int FLAGS, 11

unsigned int IP, 11

unsigned int SI, 11

V

**void interrupt dispatcher()**, 8, 17

**void readCom()**, 9, 18

**void stop\_com\_request()**, 9, 18

**void toLowerCase(char str[BIGBUFF])**, 3, 13

**void toLowerCasex(char str[BIGBUFF])**, 7, 16

**void trim(char ary[BIGBUFF])**, 4, 13

**void writeCom()**, 9, 18

W

Windows, 2

writeCom, 9, 18