

CUPRINSUL CAPITOLULUI 3:

3. DESCRIEREA FUNCȚIONALĂ.....	3-2
3.1 FUNCȚII COMUNICARE INTER-PROCESS	3-2
3.1.1 Crearea unui eveniment nou	3-3
3.1.2 Distrugerea unui eveniment	3-4
3.1.3 Legarea la un eveniment existent	3-5
3.1.4 Detașarea de un eveniment	3-6
3.1.5 Așteptarea apariției unui eveniment	3-7
3.1.6 Crearea unui modul de date	3-8
3.1.7 Crearea unui modul de date pentru un tip de memorie specificat	3-9
3.1.8 Detașarea unui modul de memorie	3-10
3.1.9 Detașarea unui modul aflat la o adresă	3-11
3.2 FUNCȚII DE TRATARE A ÎNTRERUPERILOR	3-12
3.2.1 Introduce sau distruge divice-uri din Tabela de întreruperi rapide	3-13
3.2.2 Introduce sau șterge device-uri din tabela de întreruperi	3-14
3.2.3 Întoarcere din rutina de tratare a întreruperii	3-15
3.3 FUNCȚII DE LUCRU CU SEMNALELE	3-16
3.3.1 Instalează capcană pentru interceptarea unui semnal	3-17
3.3.2 Transmiterea unui semnal către un proces	3-18
3.4 FUNCȚII IMPLEMENTATE ÎN CADRUL APLICAȚIEI	3-19
3.4.1 Setarea unui eveniment în cadrul bibliotecii	3-20
3.4.2 Așteptarea unui eveniment în cadrul bibliotecii	3-21
3.4.3 Validarea unei comenzi din cadrul interpretorului	3-22
3.4.4 Formatarea unui șir în cadrul interpretorului	3-23
3.4.5 Citirea parametrilor interni dintr-un fișier stabilit	3-24
3.4.6 Citirea unui parametru dintr-un șir stabilit	3-25
3.4.7 Citirea unui parametru tau din cadrul unui fișier	3-26
3.4.8 Planificatorul de traiectorie	3-27

3. Descrierea funcțională

Acest capitol este o descriere a funcțiilor folosite în cadrul proiectului. Vor fi prezentate, deci, funcțiile specifice OS-9, cât și funcțiile introduse de noi în cadrul bibliotecii.

3.1 Funcții comunicare Inter-Process

În cadrul acestui paragraf sunt comentate funcțiile ce intervin în proiect, cu rolul de acces la resursele partajate (memorie) sau la cozile de evenimente:

HEADER	FUNCȚIE	COMENTARIU
events.h	<code>_os_ev_creat()</code>	Crează un eveniment
	<code>_os_ev_delete()</code>	Șterge un eveniment
	<code>_os_ev_link()</code>	Se atașează la un eveniment
	<code>_os_ev_unlink()</code>	Se detașează din coada unui eveniment
	<code>_os9_ev_wait()</code>	Așteaptă să apară un eveniment
module.h	<code>_os_datmod()</code>	Crează un modul de date
	<code>_os_mkmodule()</code>	Crează un modul de date pentru un tip de memorie specificat
	<code>_os_link()</code>	Se atașează la un modul de date existent
	<code>_os_unlink()</code>	Se detașează de un modul de date existent

3.1.1 Crearea unui eveniment

```
_os_ev_creat(  
)
```

SYNTAXA:

```
#include <events.h>  
# include <types.h>
```

```
error_code _os_ev_creat(int32 winc, int32 sinc,  
u_int32 perm,  
event_id *ev_id, char *name, int32 value, u_int32  
color);
```

COMPATIBILITATE

o ANSI • OS-9 • OS-9000

STATUS

• User • System

```
DESCRIERE  
:
```

_os_ev_creat() permite crearea dinamică de evenimente. Când este creat un eveniment este specificată valoarea inițială, cât °i incrementul ce trebuie aplicat ori de câte ori evenimentul este în coada de a°teptare.

Chemările subsecvente de "_os_event" sunt folosite pentru a returna numărul de ID pentru a putea referi evenimentul creat.

"winc" specifică valoarea de auto-increment pentru "_os_ev_wait()". Numai cei mai puțin semnificativi biți din "winc" sunt folosiți.

"sinc" specifică valoarea de auto-increment pentru "_os_ev_signal()". Numai cei mai puțin semnificativi biți din "sinc" sunt folosiți.

"perm" specifică drepturile de acces. Numai cei mai puțin semnificativi biți din "perm" sunt folosiți.

```
NOTA:
```

OS-9 ignoră perm.

"ev_id" este pointer către locația în care "_os_ev_creat()" va stoca identificatorul.

"name" este un pointer către °irul care conține numele evenimentului.

"value" specifică valoarea inițială pentru variabila eveniment.

"color" specifică tipul de memorie pentru structura de evenimente.

```
NOTA:
```

OS-9 ignoră color.

Library:

os_lib.l

Pentru mai multe informații se pot consulta manualele OS-9,

3.1.2 Ștergerea unui eveniment

<code>_os_ev_delete</code> <code>()</code>
--

SYNTAXA:

```
# include <events.h>
#include <types.h>
```

```
error_code _os_ev_delete(char *name);
```

COMPATIBILITATE

o ANSI • OS-9 • OS-9000

STATUS

• User • System

DESCRIERE :

"_os_ev_delete()" șterge un eveniment din tabela de evenimente a sistemului, eliberând intrarea pentru a putea fi utilizată de alte evenimente în continuare. Un eveniment nu poate fi șters decât atunci când numărul de atașări (link-uri) este zero.

"name" este un pointer către șirul care conține numele evenimentului.

Library:

os_lib.1

Pentru mai multe informații se pot consulta manualele OS-9

3.1.3 Atașarea la un eveniment existent

_os_ev_link()**SYNTAXA:**

```
#include <events.h>
#include <types.h>
```

```
error_code _os_ev_link(char *name, event_id *ev_id);
```

COMPATIBILITATE

o ANSI • OS-9 • OS-9000

STATUS

• User • System

DESCRIERE
:

"_os_ev_link()" determină numărul de ID pentru un eveniment deja existent. Pentru a putea avea o sincronizare corectă după utilizare este de recomandat să se folosească "_os_ev_unlink()".

"name" este un pointer către șirul care conține numele evenimentului.

"ev_id" este pointer către locația în care "_os_ev_link()" va stoca identificatorul.

Library:

os_lib.1

Pentru mai multe informații se pot consulta manualele OS-9

3.1.4 Detașarea de un eveniment

<code>_os_ev_unlink</code> <code>()</code>
--

SYNTAXA:

```
# include <events.h>
#include <types.h>
```

```
error_code _os_ev_unlink(event_id ev_id);
```

COMPATIBILITATE

o ANSI • OS-9 • OS-9000

STATUS

• User • System

DESCRIERE :

`_os_ev_unlink0` decrementează contorul de utilizare al evenimentelor. Când acest contor ajunge la valoarea zero, evenimentul poate fi șters cu ajutorul funcției `_os_ev_delete()`. "ev_id" identifică evenimentul.

Library:

os_lib.l

Pentru mai multe informații se pot consulta manualele OS-9

3.1.5 Așteptarea apariției unui eveniment

<code>_os9_ev_wait())</code>
--

SYNTAXA:

```
#include <events.h>
#include <types.h>
```

```
error_code      _os9_ev_wait(event_id  ev_id,   int32
*value,
    int32 min_val, int32 max_val);
```

COMPATIBILITATE

o ANSI • OS-9 o OS-9000

STATUS

• User • System

DESCRIERE :

"_os9_ev_wait()" așteaptă apariția unui eveniment. Funcția așteaptă până când apare un eveniment, care setează valoarea în domeniul de valori specificat în comandă prin valorile de min_val și max_val. Apoi valoarea de auto-increment a wait-ului se adună la variabila de eveniment

"ev_id" specifică evenimentul.

"value" este un pointer către locația unde "_os9_ev_wait()" va stoca actuala valoare a evenimentului.

"min_val" specifică valoarea minimă de activare.

"max_val" specifică valoarea maximă de activare.

Dacă un proces din coada de așteptare a unui eveniment primește un semnal, el este activat chiar dacă evenimentul nu a apărut.

Library:

os_lib.1

Pentru mai multe informații se pot consulta manualele OS-9

3.1.6 Crearea unui modul de date

_os_datmod()**SINTAXA:**

```
#include <module.h>
#include <types.h>
```

```
error_code _os_datmod(char *mod_name, u_int32 size,
    u_int16 *attr_rev, u_int16 *type_lang, u_int32
perm,
    void **mod_exec, mh_data **mod_head);
```

COMPATIBILITATE

o ANSI • OS-9 o OS-9000

STATUS

• User • System

DESCRIERE
:

_os_datmod() crează un modul de date cu atributele specificate în parametrii funcției, °tergând porțiunea de memorie alocată modulului. Modulul este creat °i introdus în directorul de module, adică într-un director care conține toate numele modulelor din memorie.

"mod_name" este un pointer către °irul care conține numele modulului de date.

"size" este lungimea porțiunii de date.

Atributele °i codul de revizie sunt transmise funcției prin adresele locațiilor date de "attr_rev".

Tipul modulului °i numele limbajului sunt transmise cu ajutorul locațiilor adresate de "type_lang".

"perm" specifică drepturile de acces la modul. Doar primii 16 biți sunt folosiți.

"mod_exec" este adresa la care se va inițializa modulul de date.

"mod_head" este adresa la care se va seta adresa headerului modulului de date.

NOTA:

Trebuie lucrat foarte atent cu headerul modulului pentru a nu altera datele din el, astfel încât modulul să fie necunoscut sistemului.

NOTA:

Modulul creat va conține cel puțin lungimea de date dată în parametrii, dar nu este exclus ca aceasta să fie mai mare, datorită rotunjirilor datorate accesului la memorie.

Library:

os_lib.l

Pentru mai multe informații se pot consulta manualele OS-9

3.1.7 Crearea unui modul de date pentru un tip de memorie specificat

```
_os_mkmodule(  
)
```

SYNTAXA:

```
#include <module.h>  
#include <memory.h>  
#include <types.h>
```

```
error_code _os_mkmodule(char *mod_name, u_int32 size,  
    u_int16 *attr_rev, u_int16 *type_lang, u_int32  
perm,  
    void **mod_exec, mh_com **mod_head, u_int32  
color);
```

COMPATIBILITATE

o ANSI • OS-9 o OS-9000

STATUS

• User • System

```
DESCRIERE  
:
```

"_os_mkmodule()" crează un modul de date cu nume specificat și cu atribute de descriere a memoriei. Modulul este creat cu un CRC valid și este introdus și în directorul de module. În cadrul său se specifică și tipul/limbajul modulului, cât și "culoarea" pe care o are memoria suport.

"mod_name" este un pointer către irul ce conține numele modulului.

"size" este lungimea porțiunii de date.

"attr_rev" este un pointer către locația de memorie, unde "_os_mkmodule()" salvează atributele și codul de revizie al modulului.

"type_lang" este un pointer către locația de memorie, unde "_os_mkmodule()" salvează tipul de limbaj din cadrul modulului.

"perm" specifică tipul de acces la modul pentru celelalte programe. Doar cei mai puțin semnificativi 16 biți sunt utilizați.

"mod_exec" este pointer către o locație de memorie, unde "_os_mkmodule()" salvează adresa modulului de date.

"mod_head" este un pointer către locația de memorie, unde "_os_mkmodule()" salvează adresa headerului modulului de date.

"color" este tipul de memorie: SYSRAM, VIDEO1, sau VIDEO2. Dacă "color" este zero, înseamnă că nici un tip de memorie nu este specificat.

```
NOTA:
```

Modulul creat întotdeauna începe cu offsetul \$34 în modul. Acest lucru înseamnă că această funcție nu poate să creeze module, trap handlers, file managers, device drivers, sau device descriptors.

Library:

os_lib.l

3.1.8 Atașarea la un modul de memorie

_os_link()**SYNTAXA:**

```
#include <module .h>
#include <types.h>
```

```
error_code    _os_link(char    **mod_name,    mh_com
**mod_head,
    void    **mod_exec,    u_int16    *type_long,    u_int16
*attr_rev);
```

COMPATIBILITATE

o ANSI • OS-9 • OS-9000

STATUS

• User • System

DESCRIERE
:

"_os_link()" caută în directorul curent de module (sau în alternativa la directorul de module în cazul OS-9000) modulul al cărui nume, tip și limbaj se potrivește cu parametrii din funcție.

"mod_name" este numele modulului.

"mod_head" este pointerul către locația de memorie, unde "_os_link()" salvează adresa headerului modulului.

"mod_exec" este pointerul către localpia de memorie, unde "_os_link()" salvează adresa de execuție modulului.

"attr_rev" este un pointer către locația de memorie, unde "_os_link()" se uită pentru atributele și codul de revizie al modulului.

"type_lang" este un pointer către locația de memorie unde "_os_mkmodule()" se uită pentru tipul de limbaj din cadrul modulului.

Library:

os_lib.1

Pentru mai multe informații se pot consulta manualele OS-9

3.1.9 Detașarea de un modul de memorie

_os_unlink()**SYNTAX:**

```
#include <module.h>
#include <types.h>
```

```
error_code _os_unlink(mh_com *mod_head);
```

COMPATIBILITATE

o ANSI • OS-9 • OS-9000

STATUS

• User • System

DESCRIERE
:

"_os_unlink()" informează sistemul cum că procesul care a apelat funcția nu mai are nevoie de modulul de memorie și deci decrementează contorul cu numărul de atașări.

"mod_head" este un pointer către headerul modului.

NOTE:

Unele module nu pot fi detașate, spre exemplu device driverele în lucru sau modulele incluse în bootfile.

Library:

os_lib.1

Pentru mai multe informații se pot consulta manualele OS-9

3.2 Funcții de tratare a întreruperilor

Acestea sunt descrise în diferite fișiere header după cum arată tabelul următor:

FUNCȚIA	DESCRIEREA	FIȘIERUL HEADER
irq_change() ()	Setează nivelul întreruperi	reg386.h/reg68k.h
irq_disable() ()	Maschează întreruperea	
irq_enable() ()	Nemaschează întreruperea	
irq_maskget() ()	Maschează întreruperea. Se întoarce în nivelul anterior	
irq_restore() ()	Restaurează nivelul întreruperii	reg386.h/reg68k.h
irq_save() ()	Întoarce nivelul întreruperii	reg386.h/reg68k.h
_os_firq() ()	Adaugă sau șterge device-uri din tabela rapidă de întreruperi	reg386.h/reg68k.h
_os_irq() ()	Adaugă sau șterge device-uri din tabela de întreruperi	reg386.h/reg68k.h
_os9_irq() ()	Adaugă sau șterge device-uri din tabela de întreruperi	reg.h
_os_rte() ()	Întoarcere din execuție IRQ	signal.h

Funcțiile pe care le utilizăm în cadrul proiectului sunt mai puține, nefiind nevoie de toate aceste funcții date în tabelul de mai sus.

3.2.1 Introducerea sau °tergerea de divice-uri din tabela de întreruperi rapide

_os_firq()

SYNTAXA:

```
#include <reg.h>
```

```
error_code _os_firq(u_int32 vector, u_int32 priority,
void *irq_entry,
void *statics);
```

COMPATIBILITATE

o ANSI • OS-9 • OS-9000

STATUS

o User • System

DESCRIERE
:

_os_firq() instalează sau distruge o rutină de întrerupere din cadrul tabelii de întreruperi.

"vector" specifică numărul vectorului.

"priority" specifică prioritatea. "priority" este rezervată sistemului °i trebuie să fie zero.

"irq_entry" este pointer către rutina de tratare a întreruperii. Dacă aceasta are valoarea zero, rutina de tratare a întreruperii este °tearsă.

"statics" este pointer către global static storage.

Această rutină de IRQ fast, are răspunsul mai rapid decât cele aflate în schema de polling oferită de funcția _os9_irq().

Pentru a plasa o rutină de întrerupere într-un sistem de întreruperi rapide, trebuie setată intrarea din întrerupere către o valoare non-zero.

Pentru a o °terge trebuie setată intrarea din întrerupere către o valoare de zero.

NOTA:

Numai o singură rutină de _os_firq() poate fi activă pentru fiecare vector. Încercarea de a instala o a doua rutină este penalizată cu eroarea EOS_VCTBSY.

Dacă mai multe device-uri trebuie să se afle pe acela°i vector de întrerupere, atunci trebuie utilizată funcția _os9_irq() pentru a le instala.

Atentie:

Stiva oferită inițial de sistem este foarte mică. Serviciul de tratare al întreruperii trebuie să aibe grijă să nu depă°ească aceste limite.

Library:

os_lib.l

Pentru mai multe informații se pot consulta manualele OS-9

3.2.2 Introducerea sau °tergerea de device-uri din tabela de întreruperi

_os9_irq()

SYNTAXA:

```
#include <machine/reg.h>
#include <types.h>
```

```
error_code      _os9_irq(u_int32  vector,    u_int32
priority, void *irq_entry,
void *statics, void *port);
```

COMPATIBILITATE

o ANSI • OS-9 o OS-9000

STATUS

o User • System

DESCRIERE

_os9_irq() instalează o rutină de tratare a întreruperii în tabela de întreruperi a sistemului.

"vector" specifică numărul întreruperii. Numai cei mai puțin semnificativi 16 biți sunt luați în considerare.

"priority" specifică prioritatea. Numai cei mai puțin semnificativi 16 biți sunt luați în considerare.

Dacă "priority" este 0, rutina este prima din vectorul de polling.

Dacă "priority" este 255, rutina este ultima din vectorul de polling.

"irq_entry" este un pointer către începutul rutinei de întrerupere. Dacă acesta este 0, apelul va °terge întreruperea.

"statics" este pointer către global static storage. "statics" trebuie să fie unic pentru fiecare device.

"port" este adresa portului.

NOTA:

Pentru mai multe informații se pot studia manualele tehnice OS-9 Input/Output Technical Manual.

Library:

os_lib.1

Pentru mai multe informații se pot consulta manualele OS-9

3.2.3 Întoarcerea din rutina de tratare a întreruperii

_os_rte()**SYNTAXA:**

```
#include <signal.h>
#include <types.h>
```

```
error_code _os_rte(void);
```

COMPATIBILITATE

o ANSI • OS-9 • OS-9000

STATUS

• User • System

DESCRIERE

: _os_rte() termină precesarea unui semnal de întrerupere, continuând execuția codului programului curent.

Library:

os_lib.1

Pentru mai multe informații se pot consulta manualele OS-9

3.3 Funcții de lucru cu semnalele

Acestea sunt descrise în fișierul header `signal.h` după cum arată tabelul următor:

FUNCȚIA	DESCRIEREA	FIȘIERUL HEADER
<code>kill()</code>	Trimite un semnal	<code>signal.h</code>
<code>_os_intercept()</code>	Interceptează un semnal	
<code>_os_send()</code>	Transmite un semnal	
<code>_os_sigmask()</code>	Maschează un semnal	
<code>_os_sleep()</code>	Trece un proces în stare de sleep	
<code>_os9_sleep()</code>	Trece un proces în stare de sleep (standard nou)	

Funcțiile pe care le utilizăm în cadrul proiectului sunt mai puține, nefiind nevoie de toate aceste funcții date în tabelul de mai sus.

3.3.1 Instalarea unei ``capcane`` pentru interceptarea unui semnal

_os_intercept
()

SYNTAXA:

```
#include <signal.h>
#include <types.h>
#include <cglob.h>
```

```
error_code _os_intercept (void (*func)(), void
*data_ptr);
```

COMPATIBILITATE

o ANSI • OS-9 • OS-9000

STATUS

• User • System

DESCRIERE
:

_os_intercept() instalează o rutina de interceptare a unui semnal. Rutina de interceptare a semnalului este procesată asincron, pentru că un semnal (signal) poate fi trimis la orice moment de timp, fiind similar cu o întrerupere soft. Codul semnalului este transmis ca parametru pentru rutina de interceptare.

"func" este un pointer către rutina de interceptare.

"data_ptr" este un pointer către zona de stocare globală a rutinei. În general, ea este egală cu adresa ariei de date a programului.

_os_rte() este utilizată pentru ieșirea din rutina de interceptare (func). Dacă rutina nu poate folosi _os_rte() la fiecare ieșire poate rezulta un haos

NOTA:

Rutina de interceptare trebuie să fie mică și rapidă, cum ar fi setarea unui flag în zona de date.

Library:

os_lib.1

Pentru mai multe informații se pot consulta manualele OS-9

3.3.2 Transmisia unui semnal către un proces

_os_send()**SYNTAXA:**

```
#include <signal.h>
#include <types.h>
```

```
error_code  _os_send(process_id proc_id, signal_code
signal);
```

COMPATIBILITATE

o ANSI • OS-9 • OS-9000

STATUS

• User • System

DESCRIERE

:

_os_send() Trimite un semnal către un proces specificat. Un proces poate trimite semnale către toate procesele din același grup/utilizator, prin transmiterea variabilei 0 în câmpul proc_id.

"proc_id" este numărul de ID al procesului care trebuie să primească semnalul.

"signal" specifică codul semnalului ce trebuie transmis.

Library:

os_lib.1

Pentru mai multe informații se pot consulta manualele OS-9

3.4 Funcții implementate în cadrul aplicației

În cadrul proiectului de construcție a unei biblioteci de programe, s-au dezvoltat câteva funcții care sunt oarecum standard pentru lucrarea noastră, ele fiind folosite în mod curent în cadrul proiectului. Acestea fac obiectul prezentării acestui paragraf și sunt prezente în tabelul de mai jos urmând să fie prezentate mai în detaliu:

FUNCȚIA	DESCRIEREA	FIȘIERUL HEADER
evntset()	Setează un eveniment cu o valoare dată	EVNTCTRL\evntset.c
evntwait()	Așteaptă un eveniment cu o valoare dată	EVNTCTRL\evntwait.c
cmdchk()	Verifică calitatea comenzii	CMDINT\cmdchk.c
cmdfrm()	Formatează comanda	CMDINT\cmdfrm.c
getintparm()	Citește parametri interni dintr-un fișier dat	MCINIT\getintp.c
getsintparm()	Citește un șir dintr-un fișier dat	MCINIT\getsintp.c
gettauparm()	Găsește și citește parametrul tau	MCINIT\gettaup.c
tp	Trajectory-planer sau planificatorul de traiectorie	T_PLAN\tp_main.c

3.4.1 Setarea unui eveniment în cadrul bibliotecii

evntset()**SYNTAXA:**

A se compila fișierul sursă înaintea fișierului care-l folosește.

EVNTCTRL\evntset.c

```
void evntset(event_id semid, int32 set);
```

DESCRIERE

evntset() setează un eveniment cu o anumită valoare.

"semid" este ID-ul evenimentului pe care dorim să-l setăm.

"set" este valoarea pe care dorim să o ia evenimentul pe care-l setăm.

Funcția este folosită în sursele care folosesc cozi de așteptare și care lucrează cu evenimente.

3.4.2 Așteptarea unui eveniment în cadrul bibliotecii

evntwait()**SYNTAXA:**

A se compila fișierul sursă înaintea fișierului care-l folosește.

```
EVNTCTRL\evntwait.c
```

```
void evntwait(event_id semid, u_int32 wait);
```

DESCRIERE
:

evntwait() așteaptă o valoare pentru un eveniment.

Această valoare trebuie să fie sau zero sau mai mare decât 129. În cazul în care este cuprinsă între 1 și 128 înseamnă că evenimentul transmite un cod de eroare, iar funcția tipărește codul de eroare transmis de valoarea evenimentului.

"semid" este ID-ul evenimentului pe care dorim să-l citim.

"wait" este valoarea pe care dorim să o primim de la evenimentul pe care-l citim.

Funcția este folosită în sursele care folosesc cozi de așteptare și care lucrează cu evenimente.

3.4.3 Validarea unei comenzi din cadrul interpretorului

cmdchk()**SYNTAXA:**

A se compila fișierul sursă înaintea fișierului care-l folosește.

CMDINT\cmdchk.c

```
int cmdchk(char *cmd);
```

DESCRIERE
:

cmdchk() verifică compatibilitatea formatului unei comenzi în sensul că ea trebuie să aibe un head de 3 caractere și un tail.

"cmd" este șirul pe care-l transmitem funcției, pentru a fi verificat.

Funcția întoarce un cod de eroare în caz de nereușită.

Funcția este folosită acolo unde a fost nevoie de verificarea formatului unei comenzi, și anume în interpretorul de comenzi.

3.4.4 Formatarea unui °ir în cadrul interpretorului

cmdfrm()**SYNTAXA:**

A se compila fi°ierul sursă înainte de fi°ierului care-l folose°te.

CMDINT\cmdfrm.c

DESCRIERE
:

```
int cmdfrm(char in[255], char out[255]);
```

cmdfrm() formatează un °ir în a°a fel încât să poată fi prelucrat de către funcția cmdchk().

"in" este °irul pe care-l transmitem funcției pentru a fi formatat.

"out" este °irul pe care-l primim ca °ir format.

Funcția întoarce un cod de eroare în caz de nereu°ită.

Funcția este folosită acolo unde a fost nevoie ca un anume °ir să fie prelucrat, astfel încât să îndeplinească anumite condiții °i anume:

I. să nu aibe spațiu la început;

II. să nu aibe spații între primele trei caractere din comandă;

III. să aibe un singur spațiu după primele trei caractere, în cazul în care mai are o valoare după comandă;

IV. sau sa nu mai aibe spații după comandă, dacă aceasta este doar de trei caractere;

V. să nu aibe spații în cadrul valorii sau după.

3.4.5 Citirea parametrilor interni dintr-un fișier stabilit

getintparm()**SYNTAXA:**

A se compila fișierul sursă înainte de fișierul care-l folosește.

MCINIT\getintparm.c

```
void getintparm(char *intparm_file, float  
intparm[3][DOF])
```

DESCRIERE
:

getintparm() citește valorile pentru parametri interni Aa, Ad și Vmax pe care le întoarce într-o matrice de float-uri.

"intparm_file" este șirul ce conține numele fișierului cu valorile Aa, Ad și Vmax.

"intparm" este matricea cu valorile luate din cadrul fișierului. În cazul în care nu sunt găsite în fișier anumite valori pentru o variabilă, aceasta este inițializată cu zero și va duce la erori de calcul în cadrul modulelor care folosesc aceste date.

Funcția este folosită în cadrul modulelor de inițializare a acestor parametrii cum este **mcinit**.

3.4.6 Citirea unui parametru dintr-un șir stabilit

getsintparm()**SYNTAXA:**

A se compila fișierul sursă înaintea fișierului care-l folosește.

MCINIT\getsintparm.c

```
void getsintparm(FILE * fin, char *i, char *o);
```

DESCRIERE

getsintparm() citește valorile pentru parametrii interni determinați de șirul de intrare, pe care le întoarce într-un șir de ieșire.

"fin" este handlerul fișierului care este deschis pentru a putea fi prelucrat.

"i" este șirul de caractere pe care le caută funcția. El nu trebuie să aibă caracterul implicit de comentariu "#".

"o" este șirul de caractere pe care le returnează funcția. În cazul în care nu a fost găsit șirul, acest câmp este null și va duce la o valoare de zero pentru variabilă.

Funcția este folosită în cadrul modulelor de inițializare a acestor parametrii cum este **mcinit**.

3.4.7 Citirea unui parametru tau din cadrul unui fișier

gettauparm()

SYNTAXA:

A se compila fișierul sursă înaintea fișierului care-l folosește.

MCINIT\gettauparm.c

```
void gettauparm(char *intparm_file, float *tau);
```

DESCRIERE
:

gettauparm() citește valorile pentru parametrii interni determinați de șirul de intrare, pe care le întoarce într-un șir de ieșire.

"intparm_file" este șirul care conține numele fișierului ce trebuie să conțină parametrul tau.

"tau" este valoarea parametrului tau, întoarsă de funcție.

Funcția este folosită în cadrul modulelor de inițializare a acestor parametri cum este mcinit.

3.4.8 Planificatoarea traiectoriei

tp()**SYNTAXA:**

A se compila fișierul sursă înaintea fișierului care-l folosește.

T_PLAN\tp_main.c

```
void tp(char *shmp, event_id semid_0, event_id semid_1);
```

DESCRIERE
:

tp() este o funcție ce determină tipul de planificator pe care trebuie să-l execute interpretorul de comenzi.

"shmp" este adresa din memorie, la care se află modulul de memorie partajată și care conține tipul de mișcare dorit.

"semid_0" și "semid_1" sunt valorile ID-urilor evenimentelor pe care le folosesc funcțiile de planificare a traiectoriei, pentru a putea să comunice cu celelalte module.

Funcția este folosită în cadrul modulelor planificare a traiectoriei.