OS-9 – modułowy, wielozadaniowy system czasu rzeczywistego

- elastyczna, modułowa architektura
- 100% romowalność, praca bezdyskowa
- wielozadaniowość i wielodostępność
- podział czasu z wywłaszczaniem
- funkcje czasu rzeczywistego
- we/wy niezależne od sprzętu
- odporny na awarie system plików
- zgodność z UNIX-em na poziomie C
- dostępność języków wyższego rzędu
- narzędzia do uruchamiania programów na różnych poziomach (*User, System, Source*)

Programy użytkowe OS–9

attr - odczyt i zmiana atrybutów pliku

backup - duplikowanie dysku

binex - zamiana pliku na postać szesnastkową (S-rekordy)

build - tworzenie krótkich plików tekstowych

cmp - porównywanie plików

code - wyświetlanie szesnastkowych kodów klawiszy

copy - kopiowanie plików

count - zliczanie znaków, słów i linii w pliku

date - wyświetlanie daty i czasu

dcheck - sprawdzanie poprawności katalogu/dysku

deiniz - odłączenie urządzenia

del - kasowanie plików

deldir - kasowanie katalogów

devs - wyświetlanie tablicy zainicjowanych urządzeń wewy

dir - wyświetlanie zawartości kartoteki

dsave - kopiowanie poddrzewa katalogów

dump - szesnastkowe wyświetlanie zawartości pliku

echo - wysyłanie tekstu na ekran

edt - edytor liniowy

exbin - zamiana S-rekordów na postać binarną

fixmod - odtworzenie sum kontrolnych i CRC modułu

format - formatowanie dysków

free - wyświetlanie wolnego miejsca na dysku

grep - przeszukiwanie plików według wzorca

help - wyświetlanie informacji o komendach

ident - wyświetlanie informacji o modułach

iniz - inicjowanie urządzeń wewy

link - przyłączanie modułu w pamięci

list - wyświetlanie zawartości pliku

load - ładowanie modułów z pliku do pamięci

login - włączanie się do systemu (wielodostęp)

makdir - tworzenie katalogu

mdir - wyświetlanie kartoteki modułów

merge - łączenie plików na wyjście standardowe

mfree - wyświetlanie wolnego miejsca w pamięci

pd - wyświetlanie bieżącej ścieżki danych

pr - wyświetlanie pliku z formatowaniem

procs - wyświetlanie aktualnych procesów

qsort - szybkie sortowanie pliku w pamięci

rename - zmienianie nazwy pliku

save - składowanie modułów pamięciowych do plików

shell - powłoka - interfejs użytkownika, język komend

sh - powłoka - interfejs użytkownika, język komend

sleep - zatrzymanie procesu na zadany czas, lub do przerwania

tee - kopiowanie wejścia na kilka ścieżek wyjściowych

tmode - wyświetlanie i ustawianie parametrów terminala

touch - aktualizacja daty dostępu do pliku

tr - zamiana znaków w pliku na inne (filtr)

umacs - edytor ekranowy MicroEMACS 3.6 (mały)

emacs - edytor ekranowy MicroEMACS 3.10 (rozbudowany)

unlink - odłączanie modułów pamieciowych

xmode - zmiana parametrów urządzenia znakowego

Środowisko programowania w OS–9

Etapy tworzenia programu:

- tworzenie (edycja) źródła,
- kompilacja/asemblacja do plików relokowalnych (***ROF - relocatable object file***),
- łączenie (***link***) ROF-ów w moduł programowy,
- testowanie przy pomocy debuggera.

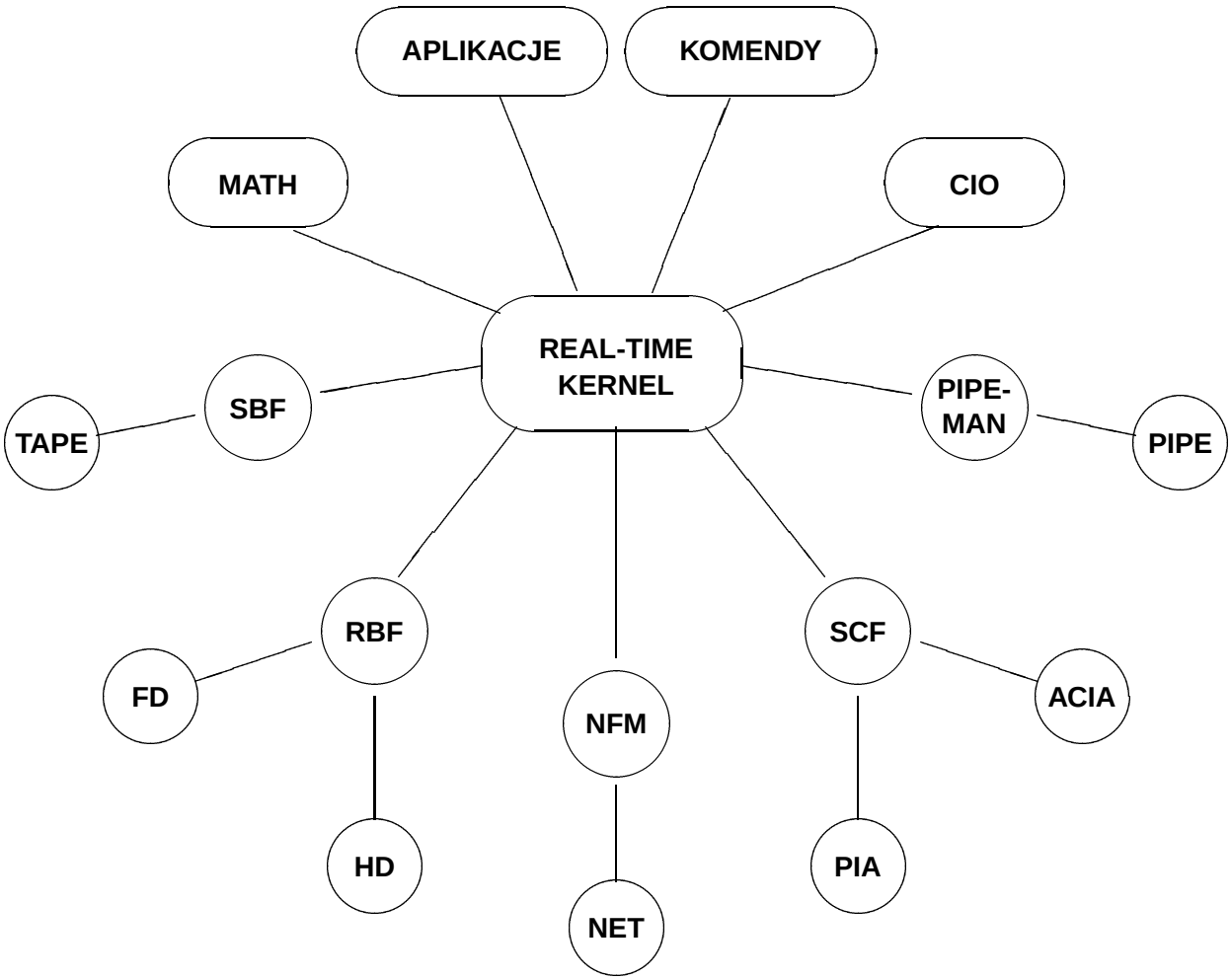
Fazy kompilacji:

- wstępne przetwarzanie (***cpp***),
- kompilacja (***c68, c68020***),
- optymalizacja (***o68***).

Narzędzia do tworzenia oprogramowania:

- ***cc*** egzekutor kompilatora C
- ***cpp*** preprocesor C
- ***c68*** kompilator C dla 68000 i 68010
- ***c68020*** kompilator C dla 68020, 68030 i68040
- ***o68*** optymizator asemblera
- ***r68*** asembler dla 68000 i 68010
- ***r68020*** asembler dla 68020, 68030 i 68040
- ***l68*** linker
- ***debug*** symboliczny debugger asemblera
- ***srcdbg*** symboliczny debugger C
- ***sysdbg*** debugger procesów w trybie systemowym

Modułowa budowa systemu OS–9



Własności modułów

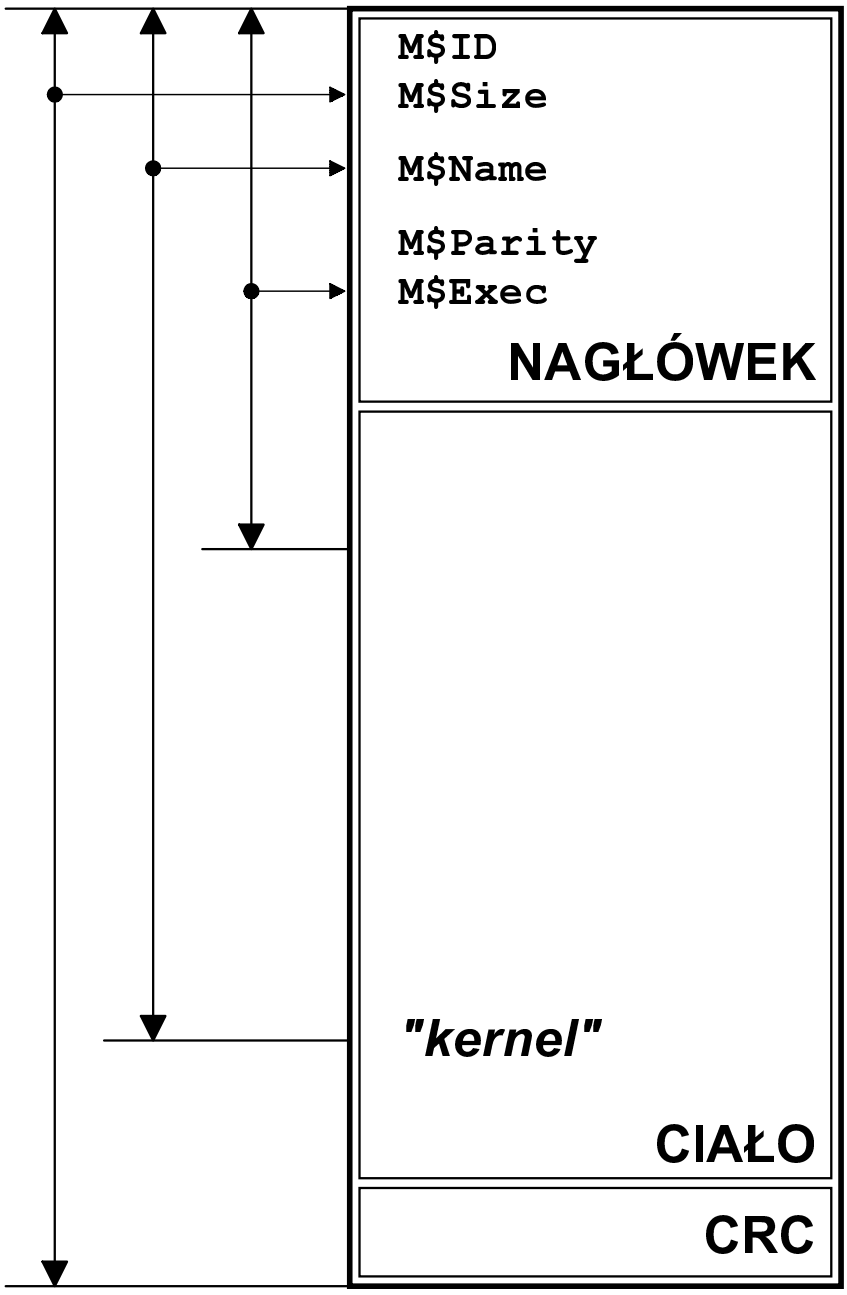
Każdy moduł musi być:

- **Re–entrant** - współużywalny, dostępny dla wielu procesów,
- **Position–independent** - niezależny od położenia w pamięci.

Moduł nie musi być:

- kompletnym programem,
- napisany w kodzie maszynowym.

Budowa modułu



Wspólna część nagłówka modułu

adr.	C *module.h*	asm *module.a*	znaczenie
00	_msync	M$ID	Znacznik modułu (4AFC)
02	_msysrev	M$SysRev	Numer wersji systemu
04	_msize	M$Size	Wielkość modułu
08	_mowner	M$Owner	Właściciel modułu
0C	_mname	M$Name	Wskaźnik nazwy modułu
10	_maccess	M$Accs	Zezwolenia na dostęp
12	_mtylan	M$Type,M$Lang	Typ i język
14	_mattrev	M$Attr,M$Revs	Atrybuty i wersja modułu
16	_medit	M$Edit	Numer edycji modułu
18	_musage	M$Usage	Wskaźnik komentarzy
1C	_msymbol	M$Symbol	Wskaźnik tablicy symboli
20	_mident		Kod identyf. modułu
22	_mspare		Zarezerwowane
2E	_mparity	M$Parity	Suma kontrolna nagłówka

Wybrane pola nagłówka

M$Accs:

```
-- -- -- -- -- pe pw pr -- ge gw gr -- ce cw cr
```

p – ***public***, **g** – ***group***, **c** – ***creator***,
w – ***write*** – pozwala na zapis do modułu,
r – ***read*** – pozwala ładować, przyłączeć i odłączać moduł,
e – ***execute*** – pozwala uruchamiać moduł.

M$Type:

1 - Prgm - moduł programowy
2 - Sbrtn - moduł podprogramów
4 - Data - moduł danych
11 - TrapLib - biblioteka
12 - Systm - składnik systemu
13 - Flmgr - moduł zarządzania plikami
14 - Drivr - moduł sterownika urządzeń
15 - Desc - moduł deskryptora urądzeń

M$Lang:

1 - **kod maszynowy**
2 - **kod pośredni Basic**
3 - kod pośredni Pascal
4 - kod pośredni C
5 - kod pośredni Cobol
6 - kod pośredni Fortran

M$Attr: `sharable sticky supervisor -- -- -- -- --`

sharable – zezwala na równoczesne używanie modułu przez wiele procesów,
sticky – moduł jest usuwany z kartoteki przy wartości MD$Link=−1, a nie 0,
supervisor – moduł pracuje w trybie uprzywilejowanym.

M$Revs:

Pozwala na dynamiczne zastępowanie istniejących modułów ich nowszymi wersjami (nawet w ROM). Jeśli ładowany moduł ma wyższy numer wersji od istniejącego w kartotece, to funkcja **F$Load** podstawia w kartotece nowy moduł w miejsce starego.

Zmienna część nagłówka modułu

Typ modułu	adr.	C	asm	znaczenie
Flmgr, Systm	30	_mexec	M$Exec	Wskaźnik startu
	34	_mexcpt	M$Excpt	Wskaźnik obsługi TRAP
Drivr	38	_mdata	M$Mem	Wielkość obszaru danych
Prgm	3C	_mstack	M$Stack	Wielkość obszaru stosu
	40	_midata	M$IData	Wskaznik inicjalizacji danych
	44	_midref	M$IRefs	Wskaznik inicjalizacji wskaznikow
TrapLib	48	_minit	M$Init	Wskaźnik inicjalizacji TRAP
	4C	_mterm	M$Term	Wskaźnik zakończenia TRAP

Systemowy katalog modułów (*Module Directory*)

adr.	asm	znaczenie
00	MD$MPtr	Adres modułu w pamięci
04	MD$Group	Identyfikator grupy modułów (adres pierwszego modułu grupy)
08	MD$Static	Wielkość pamięci zajętej przez grupę modułów
0C	MD$Link	Licznik użytkowników modułu
0E	MD$MChk	Suma kontrolna nagłówka modułu

F$Link - zwraca adres modułu o podanej nazwie, zwiększa MD$Link;

F$Load - ładuje moduł z pliku o podanej ścieżce, wykonuje F$Link;

F$UnLink - zmniejsza MD$Link dla modułu o podanym adresie, usuwa moduł i zwalnia pamięć przy MD$Link=0;

F$UnLoad - zmniejsza MD$Link dla modułu o podanej nazwie, usuwa moduł i zwalnia pamięć przy MD$Link=0.

Funkcje biblioteki C dla modułów

Funkcja C	Opis
crc()	obliczenie CRC dla modułu
_get_module_dir()	pobranie elementu kartoteki modułów
make_module()	utworzenie modułu
_mkdata_module()	utworzenie modułu danych (typu *Data*)
modcload()	załadowanie modułu do pamięci “kolorowanej” (*colored memory*)
modlink()	dowiązanie do modułu o zadanej nazwie i typie
modload()	załadowanie modułu do pamięci i dowiązanie
modloadp()	załadowanie modułu do pamięci z użyciem zmiennej *PATH* i dowiązanie
munlink()	usunięcie dowiązania do modułu o zadanym adresie
munload()	usunięcie dowiązania do modułu o zadanej nazwie

Przykład dostępu do modułu danych

```
/*
   zalozenie:
        w module typu Data o nazwie "my_module"
        sa umieszczone dane w postaci struktury
        o typie my_data;
        wskaznik dptr ma zostac ustawiony na ich
        poczatek
*/
#include <module.h>

mh_com *mhptr;  /* wskaznik struktury naglowka */
my_data *dptr;  /* wskaznik struktury danych */

/* szukanie modulu */
mhptr = modlink("my_module", 0);
/* wyjscie z bledem */
if(mhptr == -1) return (errno);
/* znalezienie wskaznika do danych */
dptr = (my_data *)((char *)mhptr + mnptr->_mexec);
```

Komendy systemowe dla modułów

mdir

```
Syntax: mdir [<opts>] [<mod names>] [<opts>]
Function: display module directory
Options:
     -a         print language instead of type
     -e         print extended directory listing
     -t=<type> list modules only of type <type>
     -u         print unformatted listing
```

load

```
Syntax: load [<opts>] {<module> [<opts>]}
Function: load a module into memory
Options:
     -d         load file from data directory
     -l         print pathlist of file loaded
     -z         get list of file names from std. input
     -z=<path> get list of file names from <path>
```

link

```
Syntax: link [<opts>] {<modname> [<opts>]}
Function: link a module in memory
Options:
     -z         get list of module names from std. input
     -z=<path> get list of module names from <path>
```

unlink

```
Syntax: unlink [<opts>] {<modname> [<opts>]}
Function: unlink modules from memory
Options:
     -z         get list of module names from std. input
     -z=<path> get list of module names from <path>
```

ident

```
Syntax: ident [<opts>] {<modname> [<opts>]}
Function: display module information
Options:
   -m           ident module in memory
   -q           quick mode, only one line per module
   -s           silent mode: quick, only disp. bad crcs
   -x           ident module in execution directory
   -z           get list of module names from std. input
   -z=<file>    get list of module names from <file>
```