RISC-OS: overview sul Kernel

di Bruno Rosati

Dopo la presentazione del Programmer's Reference Manual, apriamo da questo numero il "mini-corso" all'apprendimento del RISC-OS. Una serie di "dispense mensili" da intendersi come semplici overview informative alle quali sarà possibile accedere da tutti i livelli di utenza

Reduced Instruction Set Computer Operative System. Strutturalmente il RISC-OS si compone del Kernel — contenente le funzioni principali per la gestione del sistema — e di un insieme di moduli, per così dire addizionali che completano (semplificandone l'operatività) le potenzialità del Kernel stesso.

L'insieme di questi "moduli" è denominato System Extension Modules e in una raffigurazione "atomica" possono essere esemplificati come tanti elettroni che gravitano intorno al loro atomo. Ovvero il Kernel che per quanto riguarda questo "mini-corso" rappresenterà l'argomento centrale della prima serie di "dispense mensili": dal Character Output System (passando per l'esposizione di altre parti quali il VDU, la gestione degli Sprite, i moduli rilocabili, etc.) fino alla gestione della memoria.

Gli *elettroni*, ovvero l'insieme dei moduli di estensione — Filing System, Window Manager, il Sound System, l'FPE e via discorrendo... — saranno a loro volta analizzati successivamente. Una ripartizione ideale degli argomenti questa con la quale si cercherà di riprendere nella pratica quella che è stata appena accennata come la struttura logica del RISC-OS. In realtà il nostro modo di procedere sarà un pochino più complesso.

La prima parte (Kernel) e la seconda (System Extension Modules) invero saranno le prime ripartizioni di una più grande sezione di lavoro che chiameremo overview. In pratica, così come dice il sottotitolo, il primo ciclo del corso.

In modo opposto a quello che è il filo logico della trattazione effettuata dal Programmer's Reference Manual, tutto quello che in esso si concentra — i concetti delle chiamate SWI, quelli della Generazione degli Errori, i Vettori Software e quelli Hardware, gli Interrupts, gli Events e i Buffers — a parte qualche breve accenno, verrà quindi utilizzato solo a partire da quello che sarà un vero e proprio "Secondo Ciclo del Corso di RISC-OS".

Allorché, dall'overview iniziale e proprio grazie alla trattazione delle "routine" sopra elencate, ci porteremo ad affrontare di nuovo gli argomenti "Kernel" e "System Extension Modules" entrando nei dettagli ed approfondendone la pratica all'uso.

Attraverso l'overview familiarizzeremo, se così si può dire, con quella che

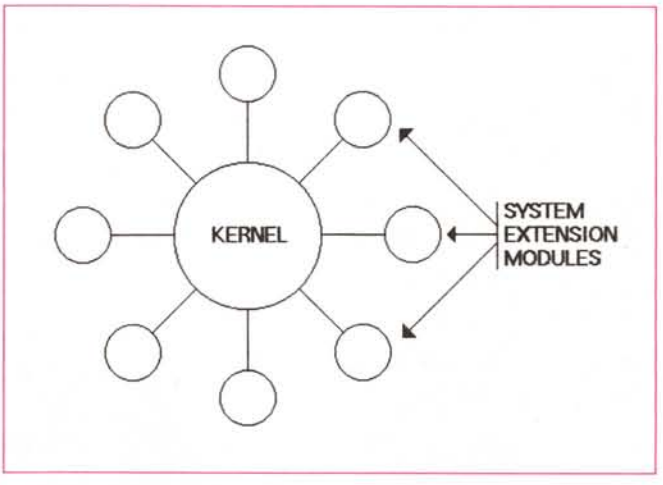

Figura 1 - Schematizzazione della struttura del RISC-OS. Dall'atomo-Kernel agli "elettroni" del System Extension Modules.

Figura 2 - In questo schema la suddivisione generale degli argomenti che andremo ad affrontare nel "Primo Ciclo" del Corso di RISC-OS.

è la struttura generale del RISC-OS, poi, anche sulla scorta delle eventuali puntualizzazioni che attraverso l'angolo dell'Archie-Mail vorrete fare, partiremo con una sorta di "apprendistato pratico".

Se qualcuno si perderà per strada torneremo a riprenderlo e se altri non avranno più voglia di sentir parlare di RISC-OS... la recensione dei vari applicativi che dalla Delphi arriveranno in redazione (e che certo non trascurerò) potrà interessarli ai propri argomenti preferiti. D'accordo?

OK; ora fate silenzio in aula.

Iniziamo la "prima lezione della prima parte del primo ciclo del primo corso di RISC-OS (!)..."

Introduzione

Tanto per cominciare potremmo anche dire che il RISC-OS, può essere considerato come l'insieme dei comandi disponibili e del CLI, il *Command Line Interface*. La semplice gestione da tastiera del sistema operativo necessita di elementari regole sintattiche da osservare e in contrapposizione ai pochi svantaggi, offre una nutrita serie di "pro".

A partire dalla facile assimilazione dell'effetto che il certo comando produce per arrivare all'elasticità di esecuzione.

Invero, la lunga serie di *** command** che si è tra l'altro sciorinato nel vecchio articolo di presentazione del RISC-OS non è la sola fonte per attingere al RISC-OS. A completare il sistema sono provviste le SWI-call. Ovvero le chiamate di tipo SoftWare Interrupts.

La prima delle quali è la OS_CLI, delegata dal RISC-OS al controllo dell'ambiente CLI. Ogni volta che si digiterà proprio un "* command", questo verrà difatti interpretato dalla OS_CLI e convertito, a secondo della sua complessità, in una o più SWI. In effetti, quando subito dopo aver digitato un comando si dà il return, l'istruzione non viene eseguita immediatamente, bensì, proprio la OS_CLI, provvederà a farne fare una traduzione utilizzando tutte le SWI di cui necessita. Solo dopo tale conversione l'operazione richiesta verrà svolta.

In pratica, quelli che noi usiamo e conosciamo come * command altro non sono che insiemi di SWI a cui viene imposta una sintassi più semplice. La "tassa" che l'uso dei * command ci obbliga a pagare è nella perdita di potenza e rapidità esecutiva rispetto alle SWI originarie.

E allora che cosa sono queste SWI, direte voi. Premesso che ne conosceremo ed utilizzeremo l'effettivo valore (analizzandone tutte le potenzialità) solo a partire dal secondo ciclo del corso, in questo primo incontro possiamo e dobbiamo comunque soddisfare la giusta fame dell'utente introducendone perlomeno il concetto.

Le chiamate in "interrupts", ideali per accedere direttamente al parco-routine del RISC-OS, in realtà sono delle istruzioni rese disponibili dall'ARM. Ogni volta che una SWI viene emessa, l'ARM va subito a verificare ad una locazione di memoria prefissata il codice del Kernel del RISC-OS con il quale verrà esaminato quale esatto tipo di SWI è ricercata e ritornata la sua esecuzione. Le SWI possono essere richiamate sia nell'ambiente dell'Assembler che in quello più amichevole del BBC-Basic.

La struttura sintattica delle SWI è molto chiara, essendo composte da un prefisso (che ne caratterizza il tipo e il modulo di appartenenza) e dal tipo di azione che eseguirà. Ad esempio, tutte le SWI con il prefisso "OS" saranno da intendersi come chiamate dirette al RISC-OS. Parlando di Character Output, più avanti si accennerà al significato della SWI chiamata OS_WriteC. Ebbene è questa il classico esempio di chiamata a RISC-OS e strutturalmente predisposta a scrivere un carattere nello stream di uscita.

Overview sul Kernel

La trattazione di questa prima parte del ciclo sarà esclusivamente espositiva: spiegare la teoria sulla quale si struttura il sistema (software) dell'Archimedes. La figura 2 schematizza per noi l'insieme dei moduli del Kernel e del System Extension ed anche se la loro ripartizione potrebbe far pensare ad una suddivisione netta e ben distinta anche a livello di sistema operativo, nell'applicabilità pratica del RISC-OS tale demarcazione non è quasi mai verificabile. Le caratteristiche dei vari moduli e sistemi all'atto pratico non sono distinguibili fra di loro.

Una suddivisione del genere è quindi più che mai stabilita proprio al fine dell'apprendimento più corretto. Quello che è Kernel e quello che è l'insieme delle estensioni.

Prendiamo allora i dieci sistemi che compongono il Kernel e vediamo che, da quello per la gestione dei caratteri (Input e Output) a quello degli Sprite, la temporizzazione relativa, le conversioni, la gestione del CLI, la funzionalità dei moduli rilocabili, il Programma di controllo e il management della memoria, ma soprattutto attraverso la valenza della Video Display Unit, il *central core* del RISC-OS è già autosufficiente per una prima gestione del computer.

A prescindere dal Filing System, il resto dell'implementato (dal Window Manager al sistema delle estensioni: Econet, Draw Module, Printer Drivers, Font Manager, etc.) possiamo considerarlo solo come la cornice di un bel quadro che, quadro, lo è già di per sé!

Il banco di lavoro offerto dal CLI, comandi e chiamate dirette all'O.S., i sistemi di streaming, controllo della tastiera e delle varie porte e periferiche, tutto catalizzato attraverso i driver del VDU. Tanto per cominciare dall'inizio prendiamo subito in esame il Character Output System.

Character Output System

Primo fra i dieci sistemi che compongono il Kernel ad essere presentato sul Programmer's Reference Manual, il sistema per l'Output dei Caratteri è quanto il RISC-OS dispone alla gestione del flusso delle informazioni verso i differenti device di uscita configurati nel

sistema. Il VDU — Video Display Unit — la Porta Seriale, i file del Filing System e l'eventuale stampante annessa al sistema stesso.

Per ciascuno di questi device, il Character Output è in grado di operare il suo controllo con altrettante modalità di lavoro, differenti fra loro e strettamente connesse a quelle che sono le caratteristiche dei singoli device.

Normalmente l'output-device (pensate ad un qualsiasi programma che avrà bisogno di visualizzare, stampare o scrivere informazioni) avviene attraverso l'uso dello Stream System — con chiamate dedicate come la OS_WriteC — il quale permette l'output di un singolo carattere alla volta bufferizzandolo in un'area della memoria a cui il programma s'indirizzerà per prelevare le informazioni contenute.

Il RISC-OS usa perciò bufferizzare tutti i device controllati dallo Stream System, ovvero il Printer Stream, il Serial Driver, lo Spool Driver e il VDU Driver. Il perché del procedimento di bufferizzazione è presto spiegato con il fatto che i programmi scritti sotto RISC-OS lavorano con un unico *rate* mentre tutti i device compresi nell'hardware controllato dal sistema agiscono su valori differenti. Quello che si verifica è una modalità operativa usualmente chiamata "asincrona"; ciò equivale a dire che programma e device non sono sincronizzati. Per non votare al "caso" il funzionamento del sistema, il RISC-OS provvede a creare delle zone di memoria dette "buffer" — ovvero: paraurti, se qualcuno vorrebbe chiamarle in italiano... — nelle quali, ciascuna per ogni device abilitato, il programma in run, depositerà le sue brave informazioni permettendo alle interruzioni di leggerle in modo asincrono.

In pratica uno speciale punto di riferimento che sia il programma che i vari device, possono raggiungere per colloquiare in modo indiretto. Tutto questo sistema di bufferizzazione viene controllato dal RISC-OS usando perlappunto lo Stream System.

Il tipo di streaming più conosciuto è senz'altro quello eseguito in fase di stampa con il **Printer Stream**, subsistema dedicato che può usare un solo device alla volta ed operante sia sulla parallela (Centronics) che sulla porta seriale; nella gestione di un network (di stampa of course) oppure convogliando le informazioni verso il Printer Driver.

Il modo comunemente usato per controllare la stampante è quello che passa attraverso le informazioni a video del VDU Driver che se disposto con il suo stream principale, una volta ricevuto il codice ASCII "chiave" (il 2, ovvero: CTRL/B) provvederà ad attivare a sua volta il VDU Printer Stream. A partire da questo momento e fino all'invio del codice contrario (ASCII uguale a 3) tutti i caratteri mandati al VDU Driver e da questo al VDU Printer Stream verranno indirizzati alla stampante.

Semplice no? In effetti semplice semplice proprio non lo è.

L'argomento difatti merita una trattazione più particolareggiata e siccome anche la SWI OS_WriteC è coinvolta in maniera determinante in questa operazione è il caso di rimandare il tutto al momento in cui provvederemo ad ampliare il discorso sul Video Display Unit e le sue enormi potenzialità.

Un passo alla volta. Per adesso ciò è più che sufficiente. Tornando all'elencazione degli altri tipi di streamer, concentriamoci sul Serial Driver e sul Serial Device. Il quale, a differenza del Printer, risulta strettamente legato allo Stream System tramite il **Serial Driver** il quale prenderà i caratteri dallo Stream System stesso per quindi introdurli nell'hardware.

Nel "blocchetto" della seriale poi avverranno tutte le modifiche che porteranno a sostituire l'informazione logica ricevuta, nell'equivalente tensione elettrica necessaria a comunicare verso l'esterno. Ciò avverrà tra l'altro in modo conforme allo standard, attraverso le linee riportate sul connettore a 9-pin che per comodità si è schematizzato in figura 3.

Il Serial Device dispone di modalità di controllo sia per la trasmissione che la ricezione dei byte-informazione. Tali controlli sono usufruibili attraverso le chiamate a disposizione del RISC-OS, il quale, a sua volta, è in grado di comunicare con la porta seriale attraverso due distinte modalità. La prima con la creazione di un buffer stream, la seconda accedendo direttamente al Serial Driver. Quest'ultima modalità sarà la più veloce, giacché verrà saltato l'intero apparato delle routine di controllo dello Stream System.

Altro device di uscita è lo **Spool Driver** dedicato al trasferimento del carattere-informazione in un file su disco sotto la gestione del Filing System. Lo spool file che verrà aperto garantirà il flusso sequenziale, carattere dopo carattere verso il file stesso.

Con il VDU Device infine si completa il quadro dei device controllati dal sistema per l'output, con i caratteri a venir diretti verso lo schermo grazie all'equivalente valore del codice ASCII che ne determinerà l'effetto. Osservando la tabella di figura 4, possiamo renderci più facilmente conto dell'effetto che appunto deriva subito dopo la verifica del valore trasmesso.

Se ad esempio verrà impartito un comando con valore ASCII compreso fra 0 e 31, il VDU Driver lo eseguirà come tale. Se l'ASCII equivalente era allora uguale a 7, il VDU sarà chiamato ad emettere un segnale sonoro, mentre se il valore, sempre esemplificando, fosse stato uguale a 16, avrebbe dovuto provvedere alla pulizia della finestra video.

Figura 3 - Serial Device. L'insieme delle connessioni (attribuzione dei valori ai pin del connettore) e l'equivalente denominazione usata dall'Acorn.

Pin	Acorn Name	Significato
1	DCD	Data Carrier Detect
2	RxD	Received Data
3	TxD	Transmitted Data
4	DTR	Data Terminal Ready
5	0v	Signal Ground
6	DSR	Data Set Ready
7	RTS	Request To Send
8	CTS	Clear To Send
9	RI	Ring Indicator

valore ASCII	Interpretazione VDU
0–31	Comandi VDU (grafici e di controllo)
32–126	Caratteri STANDARD
127	Delete
128–159	Caratteri DEFINIBILI
160–255	Caratteri INTERNAZIONALI

Figura 4 - VDU Device. La tabella esemplificativa di come ad ogni valore ASCII, il VDU Driver è in grado di rispondere producendone l'effetto relativo a video.