a cura di Corrado Giustozzi

Chorus Systèmes, System V goes microkernel

Nei sistemi distribuiti i concetti di spazio ed efficienza vanno ripensati in funzione della rete, che permette di localizzare le sezioni specializzate, affrancando il singolo elaboratore dall'elefantiasi di sistemi omnicomprensivi

di Leo Sorge

Monolitico + client/server = microkernel + servizi: è questa l'equazione sulla quale si fonda la rielaborazione dei sistemi operativi di oggi, sempre più distribuiti, ed Unix ne è stato precursore. È questa la strada seguita da diversi percorsi di ricerca, dei quali ne sono giunti sul mercato già due, Mach di Carnegie Mellon University, Nucleus di Chorus, scelti rispettivamente da Osf per la versione 1.3 e da Usl per il successore di SVR4.2.

Nel primo di questa serie di articoli, su MC 125 di gennaio, abbiamo tratteggiato una visione prospettica dei sistemi operativi dal punto di vista dell'utente, che tramite l'interfaccia grafica accede alle funzionalità del kernel e della connectivity, sia questa interna al sistema, scambio di dati o in rete. Dal punto di vista generale, l'ultimo numero chiude la trattazione sull'interfaccia utente di Unix, che è quella specificata dal consorzio Cose. Le problematiche del kernel sono state esplicitamente trascurate, in quanto pesanti da discutere e assolutamente ininfluenti alla comprensione del sistema operativo, che decidemmo di destinare ad altre trattazioni.

Stavolta, dovendo parlare di evoluzioni del kernel in forma compatta, non possiamo saltare una seppur sommaria descrizione di nucleo di sistema operativo, per vedere come viene reimplementato. Tra le due versioni di Cmu e Chorus, anche se è più anziana la prima, la seconda offre due vantaggi per ora incolmabili: una notevole presenza sul mercato e la compatibilità con System V. Mach sarà invece implementato in Osf 1.3, oggi atteso per la metà del 1994, per cui se ne parlerà in seguito.

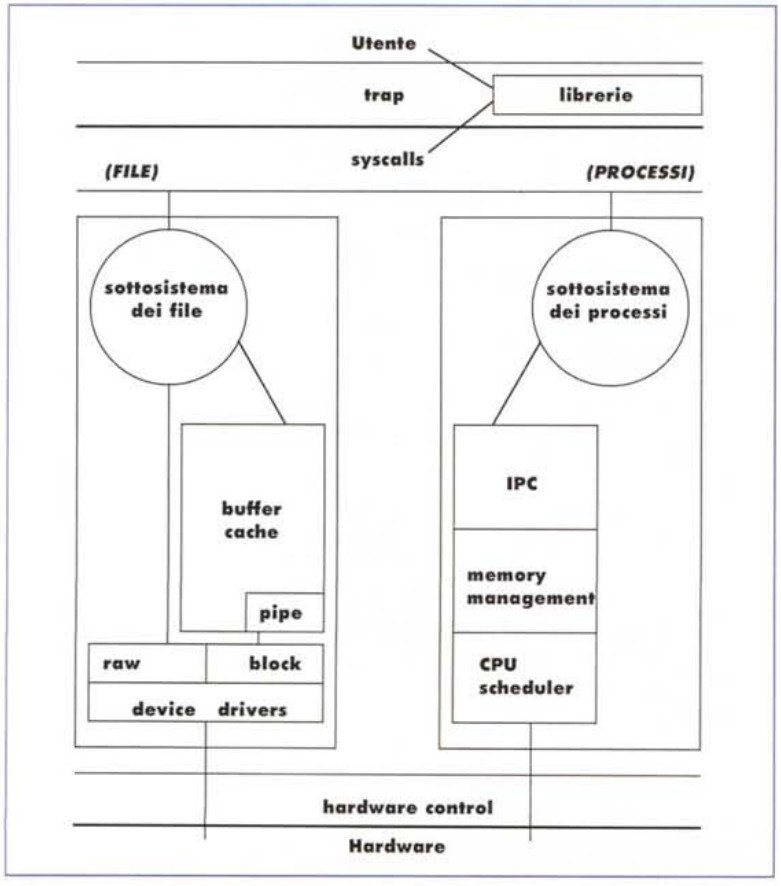


Figura 1 – Il tradizionale schema del kernel di Unix in versione monolitica, con due sottosistemi per file e processi, i pipe e i driver per periferiche, tutti imperniati sui meccanismi interni dei processi, ovvero Ipc, scheduler e memory manager.

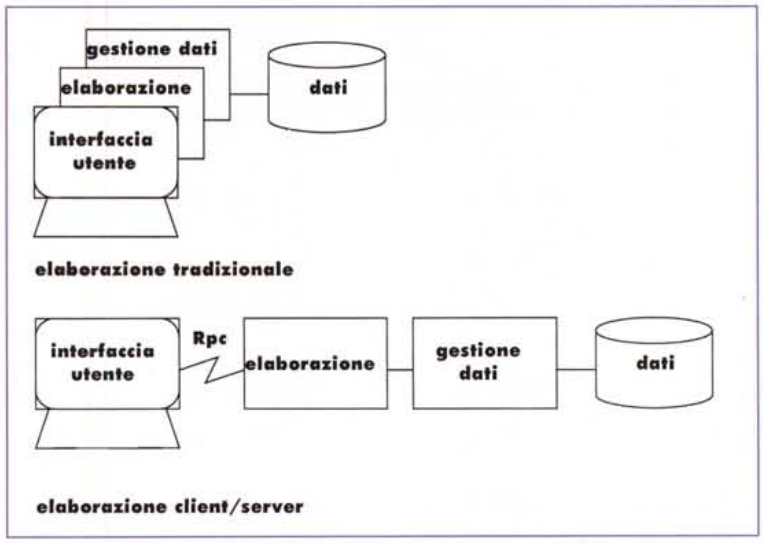
Kernel monolitici

I concetti base di questo capitolo sono: file, processi, supervisore, chiamate di sistema.

La struttura interna di Unix, o **kernel** (fig. 1) si occupa della gestione di alcune entità, delle quali le più importanti sono i **file** e i **processi**; altre funzioni, non certo secondarie ma senz'altro meno intuitive, sono le comunicazioni interne (attenzione, non necessariamente telecomunicazioni) e la configurazione all'avvio, sulle quali sorvoleremo. Come è noto, in generale un file è un insieme di dati, mentre un processo è un'istanza di un programma in esecuzione, ovvero quella parte del programma stesso che risiede nella Cpu (processore + memoria) durante l'esecuzione: in pratica il programma viene spezzato in piccole sezioni di codice che vengono eseguite singolarmente.

Ovviamente per gestire queste entità si usa del codice, organizzato in routine tra loro dipendenti, le **chiamate di sistema** o *system calls*, che possono essere eseguite in modalità non protetta (o utente) ovvero protetta (o supervisore). Per fare qualche esempio intuitivo, la gestione d'un file prevede – tra le altre – la possibilità di creare, aprire, chiudere, leggere e scrivere un file; per quanto riguarda i processi, invece, le funzioni basilari sono l'esecuzione, l'attesa, la sincronizzazione, la terminazione e la filiazione (fork). Le chiamate di System V allo stato attuale sono circa 130, e sono tutte locali, ovvero il codice è presente sulla macchina che lancia il processo.

Con riferimento alla figura 1, identifichiamo quindi due sottosistemi, per file e processi, ciascuno suddiviso in un certo numero di blocchi: **pipe** e *device driver* da un lato, e **Ipc**, *scheduler* e *memory manager* dall'altro. In un kernel monolitico la distinzione tra modo utente e modo supervisore non è formalizzata in modo preciso: poiché ciascun processo fa svariate chiamate, l'esecuzione alterna i due modi senza nessuna regola prevedibile, generando un sistema difficile da manutenere. Una seconda caratteristica dei monolitici è che raggruppano molte funzioni, per cui sono di grandi dimensioni: se vanno usati in rete, alcune delle funzioni possono essere spostate al di fuori del kernel, e passate a server specializzati. È su questa osservazione che si basa l'architettura microkernel.



▲ *Figura 2 – Nell'elaborazione tradizionale, centralizzata, tutto avviene all'interno d'una sola macchina. Nel c/s, invece, i compiti sono divisi, e gli elaboratori comunicano in rete con meccanismi di Rpc.*

Il client/server

I concetti base di questo capitolo sono: client, server, Rpc.

Un altro concetto tradizionalmente monolitico è proprio quello dell'elaborazione, che deve essere iniziata, effettuata e completata sulla stessa macchina (fig. 2). In realtà questo procedimento richiede di avere sulla stessa macchina caratteristiche di interfaccia utente, data entry, ed elaborazione di vario tipo (calcolo intenso, input/output elevato o transazionale), il che è difficile, costoso e male articolato: meglio far sì che la richiesta avvenga da una postazione adatta al *data entry* con interfaccia grafica evoluta, e che ogni specifico tipo di elaborazione sia effettuato da una macchina dedicata. Per far questo ci vuole ovviamente una connessione in rete, sulla quale il richiedente o **client**, tipicamente un personal o una workstation, invii una richiesta di elaborazione all'elaboratore che svolge il servizio, che da

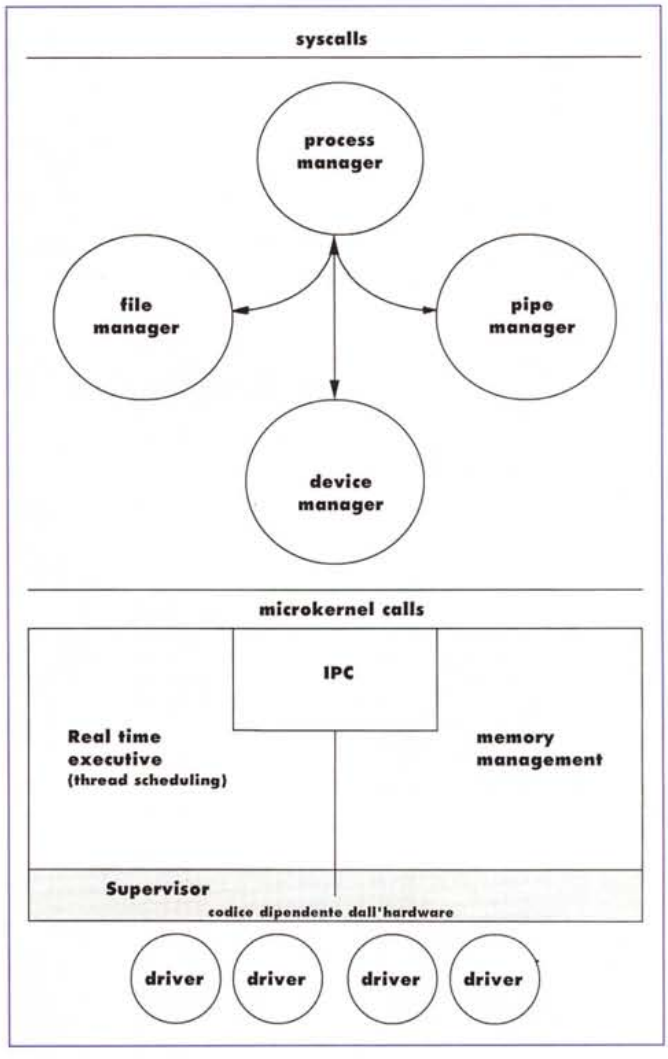


Figura 3 – La reimplementazione di System V nell'idea di Chorus. Con riferimento alla figura 1, il nucleo comprende solo Ipc, scheduler e memory manager, mentre ogni altra funzione è vista come servizio. ▶

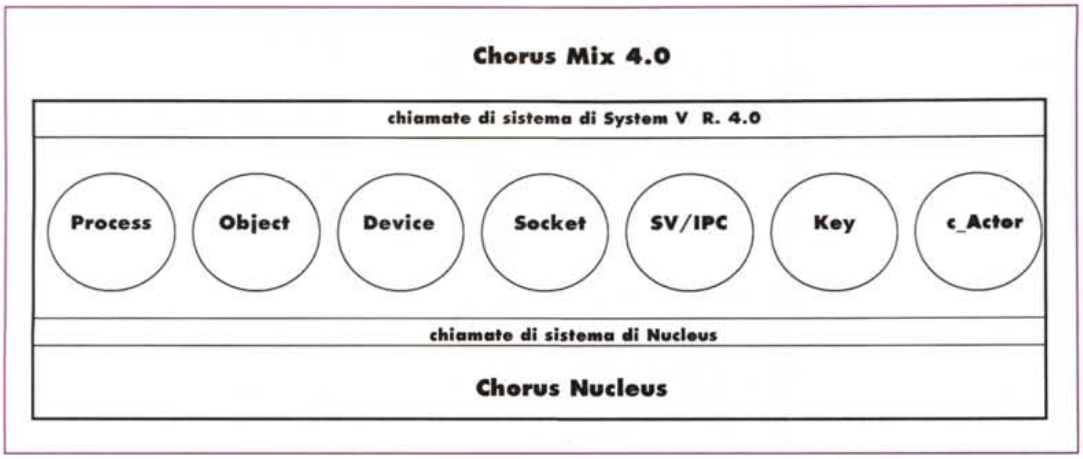


Figura 4 – Più da vicino, il nuovo sistema compatibile con SVR4. Si noti la suddivisione tra server, rappresentati da cerchi, e nucleo, con la conversione delle chiamate di sistema.

ciò trae il nome di **server**. In pratica il client non sa che l'elaborazione avviene altrove, perché la richiesta stessa ha la forma d'una tradizionale chiamata di sistema, per quanto la sintassi sia leggermente complicata per la connessione in rete, da cui il nome di *Remote procedure call* o **Rpc**.

Nell'ufficio quotidiano sono molti gli esempi di server, per la stampa, per il database, per le telecomunicazioni (fax/modem/telex/e-mail), tutti metodi di gestione di oggetti ben definiti quali stampe, archivi o messaggi, ma nulla ci vieta di astrarre questo concetto per avere dei server di fasi intermedie dell'elaborazione: è questo quanto avviene nell'implementazione di un sistema operativo basato su microkernel.

Il microkernel

I concetti base di questo capitolo sono: nucleo, servizi, thread.

In realtà un sistema operativo deve sostanzialmente eseguire programmi e dati in qualche modo residenti in memoria Ram, per cui servono solo tre moduli, uno per gestire la Ram (*memory manager*), l'altro per alternare i programmi (*scheduler*), un terzo per le interazioni tra le esecuzioni (*interprocess communications*).

Se le necessità prevedono l'esecuzione di più programmi a breve successione, e se la tecnologia è sufficientemente veloce – com'è nel nostro caso – lo scheduler sarà più sofisticato, e anziché su programmi agirà su unità di elaborazione più piccole, come i processi o i **thread**. Il thread è una nuova unità di esecuzione del software, più piccola del processo: il software scritto nel rispetto delle interfacce (specificate da Posix e da X/Open) viene scomposto in sezioni che possono essere eseguite in parallelo dai vari server di sistema, senza appesantire la gestione del processo stesso.

Se il kernel è composto solo dalle tre componenti citate, le altre vanno messe al di fuori, magari verticalizzate in server specifici che si occupino dei file, dei processi, dei pipe, delle periferiche e così via. Ciascun server risiederà su una macchina a parte, anche se alcune strutture particolarmente semplici possono accorpare più server su un unico hardware. In figura 3 vediamo il passaggio da System V monolitico alla versione microkernel di Chorus, ma anche la versione di Mach è del tutto allineata, almeno nelle linee di principio.

Curiosamente si può dire che questa filosofia, che sembra nuova e rivoluzionaria, è invece la più vecchia e tradizionale, in quanto i sistemi operativi si sono sempre fondati sulle risorse di base, quali appunto gestione della memoria, comunicazione interna ed esecuzione dei programmi (siano essi interi, suddivisi in processi o parallelizzati in thread). È stata una certa tendenza all'elefantiasi, mostrata anche nelle tecniche dei microprocessori con microcodice a complicare il chip, e dei linguaggi, a deformare il concetto originale, nella convinzione che l'*optimum* fosse un sistema omnicomprensivo, ideale oggi bandito dal consesso informatico con l'avvento dei microprocessori RISC e dei sistemi microkernel.

Chi è Chorus

La storia di questa azienda francese parte da Cyclades, un progetto di networking dell'INRIA, l'Istituto Nazionale francese per le Ricerche in Informatica ed Automatica, nato nei primi anni '70 con l'obiettivo di integrare elaborazioni e telecomunicazioni in ambito OSI e con riferimento alla commutazione di pacchetto.

Nel 1980, alla fine del progetto, il team intese continuare le ricerche, e sotto la guida di Hubert Zimmermann lanciò l'idea di Chorus, con l'obiettivo di sviluppare un sistema operativo integrato nell'ambiente di comunicazione, che trasferisse dati ed elaborazione in sistemi distribuiti. Il primo prototipo data 1984, e da allora molti altri basati su Unix, che è diventato il principale veicolo di sviluppo. Alla fine del 1986, Zimmermann, Michel Gien e Marc Guillemont fondarono la Chorus Systèmes, ovviamente basata sul lavoro dell'INRIA, mentre è del 1990 la Chorus Systems Inc, fondata negli States per sviluppare e commercializzare i prodotti.

Da allora molti sono stati i contratti: tra questi con Archipel (Francia) per Unix sui transputer Inmos T4/T8/T9, Tolérance (Francia) e Tandem (USA) per Unix fault-tolerant, con Sco (USA) per Open Desktop real-time, con Icl (GB) per elaboratori paralleli, ma anche con Motorola (ne parliamo altrove nell'articolo), Gec-Plessey Telecom, Unisys ed altre. A parte va considerato il caso Alcatel, uno dei leader delle telecomunicazioni mondiali, che dopo varie prove iniziate nel 1989 ha recentemente deciso di standardizzare l'intera attività sulla tecnologia Chorus.

Per quanto riguarda il futuro, fanno testo sia la tecnologia che l'espansione di Novell. «Gli Unix System Labs hanno una quota di Chorus, ma al momento noi non abbiamo nessuna relazione con Novell», ha detto Joel Morali, direttore vendite europee per il settore telecomunicazioni; «Comunque per scelta siamo compatibili al 100% con tutte le versioni di System V, quindi lo saremo anche con UnixWare». Sulla concorrenza con Mach, il punto di vista è semplice: «Il nostro background è nel settore delle telecomunicazioni, e sinceramente in quel campo non lo vediamo come un concorrente».

I prodotti Chorus

I concetti base di questo capitolo sono: Nucleus, Mix, personality.

Com'è ovvio, la base della proposta è il microkernel, che si chiama **Nucleus**. Giunto alla versione V3R4.1, presenta una nutrita serie di caratteristiche che lo rendono adatto all'implementazione di sistemi molto diversi tra loro sia per consistenza che per hardware di destinazione: scheduler *real time*, Ipc moderno ed ottimizzato per comunicazioni asincrone ed Rpc, il supporto per oggetti distribuiti e quello per multiprocessing sia simmetrico che asimmetrico a memoria condivisa. I device driver, secondo la più recente tecnologia, sono Dlm, moduli a caricamento dinamico, quindi non inclusi nel microkernel.

Nucleus può essere la base per qualsiasi sistema operativo, che va comunque implementato, per cui l'acquirente del microkernel deve scrivere i vari server che seguono la struttura desiderata. Sullo stesso hardware possono quindi girare più sistemi operativi, ognuno dei quali viene chiamato *personality*: è quindi possibile implementare MS-DOS, MacOs, VMS, OS/2, Windows e chi più ne ha, più ne metta, dato che oggi Nucleus gira su Sparc, Motorola 68K/88K, Intel 3/486, Amd 29K e Transputer, mentre del PowerPC si riferisce altrove in quest'articolo. In particolare la stessa Chorus è legata a doppio filo con System V tramite **Mix**, che nel 1989 con la versione 3.2 è stato il primo microkernel d'un sistema multiserver le cui prestazioni fossero uguali o superiori a quelle d'un analogo sistema ma in forma monolitica. È questa la tecnologia scelta da Usl per System V, e tale valutazione non è stata toccata dalla successiva cessione degli Usl a Novell, in quanto annunci successivi hanno confermato in Mix il riferimento per le prossime versioni distribuite di System V, che comunque rimarrà compatibile anche con Mach. Implementare un sistema con il nuovo approccio significa anche mantenere la compatibilità con tutto il software esistente, il che viene ottenuto con un'attenta conversione delle vecchie chiamate di sistema in nuove chiamate, che possono essere o locali del microkernel o remote dei server.

L'approccio microkernel, suddividendo le componenti d'un opsys, ne riduce i tempi di aggiornamento e commercializzazione, semplificando le procedure di installazione, tuning e manutenzione; la modularità consente la specializzazione dei moduli e quindi l'aumento delle prestazioni complessive; per l'utente, si riducono sia i costi diretti (acquisti) ed indiretti (manutenzione) che la configurabilità del sistema.

Attualmente esistono due versioni di Mix compatibile con System V, poiché alla 3.2 si è recentemente aggiunta la 4.0, che – oltre alla compatibilità con la nuova versione di SV – comprende un nuovo server, l'*Actor Manager*, che gestisce l'esecuzione dei c_actor, parti di codice di una nuova visione del real time. Oltre a Mix sono molte altre le implementazioni di opsys a base Nucleus, e per loro è prevista la suite di validazione di conformità al prodotto originale; analoga è la situazione per Mix, la cui aderenza agli standard è verificabile con altre suite, una per ciascuna versione.

Gli standard ai quali Mix aderisce sono la System V Verification Suite (con DDI/DKI), XPG3, Ansi C, Posix 1003.1 e Fips 151-1. Va detto che ci sono alcune sezioni di System V non supportate, e sono Rfs, la compatibilità con Xenix e gli stream, oltre ad un pugno di syscalls e poco altro. MC

Leo Sorge è raggiungibile tramite MC-link alla casella MC6750 e tramite Internet all'indirizzo MC6750@mclink.it

Chorus embedded per PowerPC

In occasione del Microprocessor Forum di Monaco, svoltosi in Germania lo scorso 19 maggio, Chorus e Motorola hanno annunciato un accordo per sviluppare e commercializzare soluzioni microkernel per PowerPC di tipo on-chip, ovvero con l'intero nucleo residente sulla Rom del microprocessore. Il principale obiettivo di mercato è il settore embedded per le telecomunicazioni, i cui fornitori chiedono grande scalabilità e modularità. A latere, Chorus ha dichiarato di ricercare altri produttori di microprocessori leader per ulteriori soluzioni embedded.

L'accordo è diretta conseguenza della definitiva scelta di Usl di supportare Chorus come microkernel per System V, confermata ufficialmente il 14 giugno ma ufficiosamente nota già da molto tempo. Tra i punti formalizzati c'è lo sviluppo d'una versione specifica per PowerPC, un development kit disponibile entro quest'anno e la disponibilità d'una prima release già nel primo trimestre del '94.

PowerPC è anche di IBM, il cui Unix è Aix, oggi interamente basato su Osf. Sulla possibilità di una implementazione Chorus di Aix, fonti ufficiali riferiscono che non c'è alcun piano al riguardo.

Glossario

Client: parte del sistema informativo che chiede informazioni, ad esempio un personal in rete con il database centralizzato.

Chiamate di sistema: routine standard componenti un sistema operativo.

File: blocco di dati.

IPC, inter-process communication: pacchetto di comunicazione tra processi qualsiasi. Estende il concetto di pipe.

Kernel: nucleo del sistema operativo.

Microkernel: gestione del nucleo di un sistema operativo del quale dal kernel vengono tolte tutte le funzioni non essenziali, che vengono reimpostate come server.

Monolitico, sistema operativo: un opsys che implementa tutte le funzioni al suo interno.

Nucleus: implementazione del microkernel secondo Chorus.

Personality: nome dell'implementazione d'un sistema operativo secondo Chorus.

Pipe: meccanismo di comunicazione tra processi correlati. È il più vecchio meccanismo di Ipc.

Processo: istanza di un programma in esecuzione. In pratica quella parte di programma in esecuzione nella Cpu (processore + memoria).

Rpc, Remote procedure call: chiamate di sistema *remote*, ovvero poste non sulla macchina chiamante ma su un'altra connessa in rete.

Server: unità in rete specializzata per un compito particolare, ad esempio la gestione d'un database, della rete o anche d'un servizio di sistema operativo (processi, oggetti...).

Servizi: compiti affidati ad un server.

Supervisore (modo): modalità di funzionamento protetta da utenti non autorizzati. Nei sistemi operativi monolitici il passaggio tra modo s. e modo utente avviene secondo regole imprecise.

Thread: unità di esecuzione dei sistemi operativi moderni, frutto d'una scomposizione del processo in parti più piccole che possono essere eseguite in parallelo.