

Diario delle lezioni — Sistemi Operativi

Marino Miculan

A.A. 2003-2004

Indice

1	Introduzione ai Sistemi Operativi (29/9)	2
2	Evoluzione dei Sistemi Operativi (30/9)	3
3	Architettura degli elaboratori (6/10)	3
4	Componenti e servizi dei sistemi operativi (7/10)	3
5	Struttura dei sistemi operativi (13/10)	3
6	Processi (14/10)	4
7	Processi 2 (20/10)	4
8	Thread (21/10)	4
9	Thread 2 (27/10)	4
10	Scheduling di breve termine (28/10)	5
11	Scheduling di breve termine (3/11)	5
12	Scheduling di breve termine (4/11)	5
13	Esempi di Scheduling di breve termine; esercitazioni (10/11)	5
14	Programmazione concorrente (11/11)	5
15	Semafori (17/11)	5
16	Monitor. Problemi Classici (18/11)	6
17	IPC di Unix (24/11)	6
18	IPC di Unix: pipe, code di messaggi, segnali (25/11)	6
19	Deadlock I (14/1)	7
20	Deadlock II (15/1)	7

21 Gestione della Memoria I (15/1)	7
22 Gestione della Memoria II (21/1)	7
23 Gestione della Memoria III (22/1)	8
24 Memoria virtuale I (23/1)	8
25 Memoria virtuale II (28/1)	8
26 Memoria virtuale III (29/1)	8
27 Gestione della Memoria in Unix e Windows (29/1)	8
28 I/O 0: Riassunto di Hardware (4/2)	9
29 I/O I (5/2)	9
30 I/O II (6/2)	9
31 I/O III (11/2)	9
32 I/O IV (12/2)	9
33 I/O Unix 2 e Windows (13/2)	10
34 File system I (18/2)	10
35 File system II (19/2)	10
36 File system III (20/2)	10
37 File system IV (25/2)	11
38 Esempi di file system (26/2)	11
39 Esempi di file system (27/2)	11
40 Sistemi multiprocessore e multicomputer (3/3)	11
41 Sistemi di rete e distribuiti (4/3)	12
42 Servizi e Sistemi di rete (5/3)	12
43 Servizi e Sistemi distribuiti (10/3)	12
44 RPC, NFS (11/3)	13

1 Introduzione ai Sistemi Operativi (29/9)

Administrivia: struttura del corso, di Lab S.O., esami, bibliografia,...

Cosa è e cosa fa un sistema operativo: prime risposte. Obiettivi di un S.O.: uso efficiente delle risorse, implementazione di una macchina astratta. Posizione del S.O. in un sistema di calcolo. Evoluzione dei sistemi operativi.

2 Evoluzione dei Sistemi Operativi (30/9)

Evoluzione dei sistemi operativi, con esempi di funzionalità introdotte via via (spool, multiprogrammazione, time-sharing, multiutente,...). Trasferimento di caratteristiche dei S.O.: dalle macchine più grandi a quelle più piccole, mano a mano che quest'ultime aumentano in potenza e prestazioni.

3 Architettura degli elaboratori (6/10)

Richiami all'architettura dei calcolatori elettronici: componenti fondamentali, loro funzionamento di base.

Comunicazione CPU-dispositivi di I/O: busy-wait, polling, interrupt, DMA. Interruzioni software e hardware: definizioni, esempi, gestione (vettore di interruzione, stack,...).

Protezione hardware: istruzioni privilegiate, i due modi di esecuzione della CPU (*user* e *supervisor*). System calls.

4 Componenti e servizi dei sistemi operativi (7/10)

Componenti comuni dei sistemi operativi: Gestione dei processi, Gestione della Memoria Principale, Gestione della Memoria Secondaria, Gestione dell'I/O, Gestione dei file, Sistemi di protezione, Connessioni di rete (*networking*), Sistema di interpretazione dei comandi.

Servizi offerti dai sistemi operativi: Esecuzione dei programmi, Operazioni di I/O, Manipolazione del file system, Comunicazioni, Individuazione di errori. Addizionali: Allocazione delle risorse, Accounting, Protezione.

Implementazione dei servizi: le *system calls*. System calls: tipi, funzionalità offerte. Meccanismi di passaggio dei parametri tra codice utente e codice kernel: in registri, in tabelle puntate da registri, sullo stack.

Programmi di sistema: l'ambiente offerto insieme al sistema operativo. Tipi di programmi di sistema.

5 Struttura dei sistemi operativi (13/10)

Struttura dei sistemi operativi: struttura semplice (DOS, Unix vecchia maniera); svantaggi. Struttura stratificata (OS/2): vantaggi in modularità. Macchine virtuali: robustezza, complessità implementativa, svantaggi nella comunicazione tra processi, esempi (DOS sotto Windows, JVM,...). Exokernel.

Verso il microkernel: distinzione tra *meccanismi* e *politiche*. Idea di microkernel: spostare in user space tutte le politiche e lasciare in kernel solamente i meccanismi. Funzionamento di base di un microkernel. Vantaggi e svantaggi dei microkernel. Esempi di sistemi operativi basati su microkernel.

Struttura di Mach, Windows 2000, Mac OS X.

Caratteristiche fondamentali del sistema UNIX, principi di progetto. Struttura a strati: hardware, kernel, librerie, applicativi. Struttura del kernel: a strati (tradizionale), modulare (moderna). Interfaccia per il programmatore. Alcuni esempi di chiamate di sistema.

6 Processi (14/10)

Processi: concetto di processo, risorse. Switch di contesto. Diagramma degli stati, process control block. Scheduling: switch di contesto, code di scheduling dei processi, migrazione dei processi. Gestione delle interruzioni. Scheduling di lungo, medio e breve termine.

7 Processi 2 (20/10)

Scheduler a breve termine: controllo del time sharing. Scheduler a lungo termine: controllo del job mix. Processi CPU-bound e I/O-bound. Scheduler a medio termine: controllo della multiprogrammazione.

Modelli di esecuzione (a kernel separato, a processo che esegue codice kernel, a microkernel).

Processi in UNIX: modello di esecuzione “ad iceberg”, modello della memoria, creazione, terminazione. Ciclo fork-execve nella shell.

Strutture dei processi in Unix: process structure, user structure, text structure. Informazioni contenute. Posizione nella memoria. Real e effective UID/GID.

Cosa succede al *fork()*: cosa viene copiato, cosa no, cosa viene condiviso. Cosa viene sostituito con la *execve*.

8 Thread (21/10)

Diagramma degli stati di uno Unix tradizionale.

Thread. Concetti fondamentali: separazione tra unità di allocazione risorse e unità di esecuzione, cosa è del processo e cosa invece del thread. Esempi di utilizzo (word processor, http server).

9 Thread 2 (27/10)

Thread a livello utente (ULT) e a livello kernel (KLT). Caratteristiche, differenze, vantaggi e svantaggi. Scheduling a livello utente e a livello kernel, comportamento su macchine multiprocessore, costo di creazione/distruzione. Thread CMU, POSIX, Mach. Implementazione dei thread. Informazioni di thread e di processo. Esempi di implementazione. La soluzione Solaris: i Light Weight Process, loro utilizzo e flessibilità. La soluzione Linux: i cloni. La soluzione Windows.

10 Scheduling di breve termine (28/10)

Scheduling di breve termine: concetti base, obiettivi, parametri da ottimizzare. Diagrammi GANTT. Primi algoritmi: First Come-First Served, Shorted Job First, Shortest Remaining Time First.

11 Scheduling di breve termine (3/11)

Previsione del prossimo CPU burst. Round Robin a code multiple, con/senza feedback. Schedulazione garantita. Schedulazione a lotteria.

12 Scheduling di breve termine (4/11)

Cenni a scheduling multiprocessor: tipi (omogenei, eterogenei), pinning, problemi, soluzioni (AMP, SMP).

Sistemi real-time: differenze tra *hard* e *soft* real-time, caratteristiche dello scheduling soft real-time. Processi schedulabili. Algoritmi per realtime: Rate Monotonic Scheduling, Earlier Deadline First.

Riduzione del tempo di latenza: kernel non prelazionabile, con punti di prelazione, pienamente prelazionabile.

Lo scheduling di Unix tradizionale: code multiple round robin, con feedback. Code a priorità negativa per i processi in modalità kernel.

13 Esempi di Scheduling di breve termine; esercitazioni (10/11)

Scheduling di SVR4: principî, classi di default. Esempio: lo scheduling di Solaris. Comandi *prionctl* e *dispadmin*.

Scheduling di Linux 2.4: supporto per real-time, politiche FIFO, RR, OTHER. Cenni allo scheduling $O(1)$ di Linux 2.6.

Scheduling di Windows 2000.

Esercitazioni.

14 Programmazione concorrente (11/11)

Processi cooperanti. Vantaggi della suddivisione dei task tra più processi. Svantaggi: problemi di sincronizzazione e condivisione. Esempio: il produttore/consumatore su un buffer limitato condiviso. Race condition: definizione, caratteristiche, esempio nel produttore/consumatore sulla variabile condivisa.

Problema della sezione critica. Requisiti di una soluzione. Prime soluzioni: disabilitazione degli interrupt. Soluzioni software: alternanza stretta, algoritmo di Peterson

15 Semafori (17/11)

Algoritmo del fornaio.

Supporto hardware alla sincronizzazione. Istruzioni di Test&Set.
Evitare i cicli busy-wait: *sleep* e *wakeup*. Esempio del produttore-consumatore.
Semafori. Esempi: mutua esclusione per sezione critica, produttore-consumatore, sincronizzazione. Implementazione. Mutex. Necessità della memoria condivisa.
Deadlock e starvation: definizione.

16 Monitor. Problemi Classici (18/11)

Monitor: definizione, esempio, problematiche implementative.

Passaggio di messaggi: definizione, problematiche implementative. Comunicazione sincrona vs. asincrona. Esempio: produttore-consumatore.

Barriera.

I Grandi Problemi Classici: cinque filosofi, lettori/scrittori. Cenni al problema del barbiere.

17 IPC di Unix (24/11)

I meccanismi IPC di System V. I semafori in Unix: creazione (chiamata *semget(2)*), utilizzo (chiamata *semop(2)*). Descrizione funzionamento e caratteristiche dei semafori interi (multivalore) e multipli (array di semafori).

La memoria condivisa: creazione (chiamata *shmget(2)*) e utilizzo (chiamata *shmat(2)*). Descrizione funzionamento e caratteristiche della memoria condivisa: volatilità, protezioni. Esempio: i programmi *pline* e *cline*. Race condition possibili in *pline*, *cline*, loro risoluzione con semafori nei programmi *sempline* e *semcline*.

Implementazione alternativa: il memory mapping attraverso la chiamata *mmap(2)*. Differenze rispetto alla memoria condivisa. Esempio: i programmi *mmpline* e *mmcline*.

Monitoraggio stato IPC: i comandi *ipcs(1)* e *ipcrm(1)*.

Primitive di sincronizzazione di thread in Solaris (cenni): semafori, mutex adattativi, variabili condition, lock di lettura/scrittura.

18 IPC di Unix: pipe, code di messaggi, segnali (25/11)

Le pipe: creazione (chiamata *pipe(2)*), utilizzo. Descrizione funzionamento e caratteristiche delle pipe. Applicazioni: situazioni produttore-consumatore. Esempio: il programma *whosort.c*. Named pipe: la creazione con *mknod(2)*.

Le code di messaggi: creazione (chiamata *msgget(2)*), utilizzo (chiamate *msgrcv(2)*, *msgsnd(2)*). Descrizione funzionamento e caratteristiche.

Segnali: cosa sono, da dove vengono. I segnali POSIX. Comportamenti di un processo in seguito ad un segnale. Esempio: un interrupt handler. Settaggio del signal handler, invio dei segnali, tracing del processo. Possibili usi di un segnale: ignorarlo, uscire in maniera pulita, riconfigurazione dinamica, (dis)attivare il debugging, riportare tabelle interne, implementazione di un timeout.

19 Deadlock I (14/1)

Introduzione ai deadlock. Risorse: prerilasciabili, non prerilasciabili. Protocollo di accesso ad una risorsa non prerilasciabile. Uso di semafori mutex per rappresentare le risorse. Problemi con più risorse: attese circolari. Problema di determinare una sequenza di allocazione sicura. Politiche di scheduling safe: FSCS. Non safe: Round Robin. Problemi: riconoscere la possibilità di deadlock, riconoscere un deadlock, risolvere un deadlock.

Definizione di Deadlock. Condizioni necessarie per il verificarsi del deadlock (mutua esclusione, hold&wait, mancanza di prerilascio, catena di attesa circolare). Grafo di allocazione risorse. Uso ipotetico del grafo di allocazione risorse in uno scheduler.

Gestione del deadlock. Quattro alternative: fare niente, riconosce e curare un deadlock, evitare dinamicamente le situazioni pericolose, garantire che una delle quattro condizioni necessarie non valga.

Ignorare il problema: nel caso in cui i costi siano troppo alti.

20 Deadlock II (15/1)

Secondo approccio: identificazione e risoluzione dei deadlock. Algoritmi di identificazione per risorse singole e risorse multiple. Uso degli algoritmi di identificazione. Risoluzione dei deadlock: prerilascio, rollback, terminazione.

Terzo approccio: eludere i deadlock. Traiettorie di risorse. Stati sicuri, sequenze sicure. Algoritmo del banchiere.

Quarto approccio: prevenzione dei deadlock. Falsificare una delle condizioni necessarie.

Approccio combinato. Blocco a due fasi.

21 Gestione della Memoria I (15/1)

La gerarchia della memoria. Obiettivi della gestione della memoria. Mono-programmazione. Multiprogrammazione: Performance di utilizzo CPU, analisi della performance. Collegamento degli indirizzi di dati e istruzioni alla memoria fisica: compile time, linking time, run time. Caricamento dinamico. Collegamento dinamico. Spazio di indirizzi logici e fisici. La MMU: mappatura indirizzi virtuali in indirizzi fisici.

22 Gestione della Memoria II (21/1)

Allocazione contigua: partizione singola, partizionamento statico, partizionamento dinamico. Frammentazione interna. Algoritmi di allocazione: first-fit, next-fit, best-fit, worst-fit. Swapping.

Allocazione non contigua: paginazione, segmentazione. Esempi. Protezione e condivisione in ogni situazione. Supporto hardware necessario, traduzione degli indirizzi logici in indirizzi fisici in ogni situazione.

23 Gestione della Memoria III (22/1)

Implementazione della paginazione: hardware necessario, paginazione ad un livello, costi e degrado delle performance. Translation lookaside buffer: hardware, software. Calcolo del tempo di accesso effettivo. Principio di località. Paginazione a più livelli. Tabella delle pagine invertita. Segmentazione paginata: la soluzione MULTICS (segmentazione con paginazione a 1 livello), la soluzione IA32 (segmentazione con paginazione a 2 livelli).

24 Memoria virtuale I (23/1)

Memoria virtuale. Paginazione on demand: benefici, costi. Page fault: definizione, gestione. Sostituzione di pagina: costi, limiti architetturali. Performance. Copy-on-write, memory-mapped I/O.

Algoritmi di rimpiazzamento: FIFO (anomalia di Belady). Algoritmi di stack. Algoritmi OPT, LRU.

25 Memoria virtuale II (28/1)

NFU. Implementazioni a contatori e a stack di LRU. CLOCK e CLOCK migliorato.

Thrashing. Principio di località. Gestire il thrashing: modello del working set, limitazione del page fault.

26 Memoria virtuale III (29/1)

Implementazioni dell'algoritmo del working set. Algoritmo WSClock.

Sostituzione globale verso locale. Allocazione dei frame: libera, equa, a priorità. Free list con buffering. Prepaging. Effetti della dimensione della pagina. Influenza della programmazione sul page fault rate.

27 Gestione della Memoria in Unix e Windows (29/1)

Il caso UNIX. Quando risulta necessario liberare memoria: dopo una FORK, una BRK, uno stack overflow, un copy-on-write, o uno swap in.

Prima del 3BSD, e di SVR4, l'unico modo per liberare memoria era lo *swapping*. Dal 3BSD: la paginazione. Struttura della memoria fisica in 4BSD: kernel, mappa centrale (core map), frames.

La paginazione on demand in Unix: chi alloca le pagine (il kernel), e chi le libera: il processo 2, *pagedaemon*. Quando viene attivato, cosa fa. Algoritmo di paginazione two-handed clock. Variante con isteresi. Il processo 0 di Unix: *swapper*, o *scheduler*. Quando si attiva. Descrizione del funzionamento.

Come monitorare il funzionamento della memoria virtuale: lettura del comando *ps(1)*. , *vmstat(1M)*, *top(1)*.

Il caso Windows 2000. Modello della memoria. Stati di una pagina. Algoritmo di allocazione. Algoritmo di paginazione. Thread del kernel che si occupano della paginazione.

28 I/O 0: Riassunto di Hardware (4/2)

Tipi di dispositivi I/O. Meccanismi di comunicazione tra CPU e controller: mappatura in memoria, con spazio separato. Modi di I/O: programmed I/O, interruzioni (vettore di interrupt), DMA, DVMA.

Gestione degli interrupt. Interruzioni precise, non precise. Effetti sulle architetture pipeline e superscalari. Diverse soluzioni (in hardware, o in software, nel S.O.).

Le componenti hardware/software dell'I/O: controller, driver, sw device/independent, sw a livello utente. Evoluzione di un sistema di I/O.

29 I/O I (5/2)

Interfaccia di I/O per le applicazioni: uniformità, incapsulamento. Parametri e caratteristiche dei dispositivi. Classi di dispositivi: I/O a blocchi, a carattere, mappato in memoria, socket di rete. Casi che esulano da queste classi (timer, orologio).

I/O bloccante, non bloccante, asincrono.

Sottosistema di I/O del kernel: funzionalità richieste (scheduling, buffering, caching, spooling, accesso esclusivo, gestione degli errori).

30 I/O II (6/2)

Descrizioni di ogni strato del sottosistema di I/O. I driver degli interrupt. I driver dei controller. Lo strato device-independent (interfaccia uniforme, denominazione uniforme, bufferizzazione, gestione errori, allocazione/rilascio dei dispositivi ad accesso esclusivo). Software a livello utente: librerie (esempio: `printf`), processi di sistema (demoni).

31 I/O III (11/2)

Traduzione delle richieste di I/O in operazioni hardware: comunicazione tra gli strati all'atto di un I/O.

Performance: costi nella gestione di I/O. Esempio: il telnet. Ottimizzazioni. Livello di implementazione delle funzionalità.

Gestione dei dischi. Algoritmi di scheduling dei dischi: FCFS, SSTF, SCAN, C-SCAN, LOOK, C-LOOK. Considerazioni.

32 I/O IV (12/2)

Gestione dello spazio di swap: soluzioni 4.3 BSD, Solaris, Windows.

Affidabilità e prestazioni dei dischi: i sistemi RAID. Livello 0, 1, 4, 5.

Input/output in Unix. Obiettivi. Principali chiamate di sistema: *open(2)*, *close(2)*, *write(2)*, *read(2)*, *ioctl(2)*, *lseek(2)*, *mmap(2)*, *fsync(2)*...

Visione astratta dei device: i file in */dev*. Come leggere *ls -l /dev*: device a blocchi/a carattere, major e minor number. Ruolo dei major e minor number: tabelle dei device, parametro dei driver. Esempi. Il comando *mknod*. Protezione dei device.

Struttura del kernel I/O secondo 4.3BSD. Compiti di ogni strato. Funzionalità dello strato device-independent. Funzionalità implementate da un device driver in Linux: la struttura *file_operations*. Registrazione di un driver. Come la chiamata di libreria risale alla corrispondente routine del driver.

33 I/O Unix 2 e Windows (13/2)

Accesso *cooked* e *raw* (*canonical* e *noncanonical*, secondo POSIX). Per i dispositivi a blocchi: cache, ottimizzazione dei tempi di accesso ai dischi. Il controllo di un dispositivo a caratteri: il comando *stty(1)*.

STREAM di System V: idea di base, funzionamento, esempi.

Monitoraggio I/O: il comando *iostat(1M)* di Solaris.

Il sottosistema di I/O di Windows 2000. Caratteristiche generali. I/O Asincrono. Il *Microsoft Driver Model*. Gli I/O request packets (IRP). Il PnP manager: riconoscimento dei dispositivi, caricamento dei driver, inizializzazione dei device e strutture correlate. Stack di driver in Windows.

Esercitazioni.

34 File system I (18/2)

I file sono delle astrazioni di dati persistenti. Struttura, nomi, modi di accesso, protezione, e implementazione dei file sono peculiari del S.O. Aspetti generali. Attributi dei file: i metadati.

Operazioni sui file. Esempio di uso dei file. File mappati in memoria.

Directory: requisiti, obiettivi.

35 File system II (19/2)

Strutture tipiche di directory: piatte, ad albero, DAG, a grafo. Caratteristiche, problematiche.

Protezione: matrice di accesso, sua decomposizione in *access control list* o *capability list*. La protezione in UNIX: bit di modo, dominio del processo. Come restringere l'accesso ad un file ad un gruppo di utenti. Effective UID e GID. Setuid e Setgid bit, loro impiego.

36 File system III (20/2)

Implementazione e struttura del file system. Tabella dei file aperti. Mounting e unmounting dei file system.

Allocazione dei blocchi: contigua, concatenata, indicizzata. Vantaggi e svantaggi di ogni soluzione. Esempi: FAT16, FAT32. Inodes di Unix: Contenuto e

utilizzo di un inode: blocchi diretti, primi, secondi, terzi indiretti. Limiti massimi di un file: imposti dal file system, imposti dall'offset. Gestione dello spazio libero: bitmap, linked list.

37 File system IV (25/2)

Implementazione delle directory: a dimensioni fisse, a heap; lineari, ad hash, ad albero bilanciato.

Problemi di efficienza e performance. Influenza della dimensione del blocco. Cache. Recovery da failure: utility di ripristino della consistenza (scandisk, fsck).

38 Esempi di file system (26/2)

File system journalled, esempi.

Strutturazione di un disco fisico (partizionamento).

La FAT di Windows. Le directory in DOS e Windows.

Il file system di UNIX. File system supportati, file system nativi (UFS, EXT2, EFS, SysV). Dimensioni dei blocchi in EXT2, SysV. Blocchi e fragment in UFS. I-node.

39 Esempi di file system (27/2)

Cosa succede nel kernel quando si apre un file: caricamento dell'i-node in memoria. Lista degli i-node in core, la file structure, la tabella dei file aperti per processo. Dove risiedono queste strutture, e perché ci sono (importanza della file structure). Il comando *sync(1m)*.

Le directory: struttura di BSD (nomi fino a 255 caratteri, link simbolici...). Hard links. Traduzione pathname (relativo e assoluto) nel relativo i-node.

Esempio di file system fisico: il file system EXT2 di Linux: partizioni, bootblock, superblock, gruppi, tabella degli i-nodes, blocchi.

Collegamento del file system al logical file system: il comando *mount(1m)*. Overlay dei file system. Al boot: / è montato dal kernel, gli altri vengono montati dalla tabella */etc/fstab*. Come vengono tradotti i pathname nel caso incontriamo un i-node che sia un mount point: la tabella */etc/mnttab*. Altri file system montati con *mount*.

Usi di fsck: verifiche di integrità del file system prima del mounting, riparazione dei block link counters, recupero del superblock.

Il File System NTFS: principi di progetto, caratteristiche di base. Il Master File Table (MFT): struttura, contenuti. Esempi di record. Attributi residenti, non residenti. File: allocazione, immediati, non residenti, run. Directory: corte (liste), lunghe (B+tree).

40 Sistemi multiprocessore e multicomputer (3/3)

Introduzione ai sistemi a più processori. Processori tightly-coupled, loosely-coupled. Sistemi multiprocessore, multicomputer, distribuiti: differenze.

Sistemi multiprocessore: SMP UMA, NUMA.

Sistemi operativi per multiprocessore: locale, AMP, SMP.

Multicomputer: motivazioni, differenze con i multiprocessori. Topologie di connessione per multicomputer. Interfacce di rete. Programmazione nei multicomputer con **send** e **receive**. **send** bloccante versus non bloccante. Programmazione con RPC. Difficoltà delle RPC.

41 Sistemi di rete e distribuiti (4/3)

Il modello client/server. Motivazioni all'uso dei sistemi distribuiti: condivisione delle risorse, accelerazione della computazione, aumento dell'affidabilità, comunicazione. Confronto con gli altri modelli di calcolo parallelo.

Hardware dei sistemi distribuiti: reti LAN, WAN.

Programmazione nei sistemi distribuiti: necessità di un modello di calcolo comune. Il *middleware*.

Servizi di rete e servizi distribuiti. Esempi.

Servizi di rete: orientati alla connessione, senza connessione. Affidabilità del servizio. Esempi di diversi protocolli. Protocolli. Modello ISO/OSI. Stack TCP/IP. Principali protocolli.

42 Servizi e Sistemi di rete (5/3)

Communication endpoint per servizi di rete: le socket. Sono strumenti di IPC tra macchine remote. Tipi di socket, famiglie di indirizzamento.

Strutture dati per le socket. Chiamate di sistema *socket(3n)*, *bind(3n)*, *listen(3n)*, *accept(3n)*. Esempio di server TCP. Come gestire più utenti contemporaneamente: server forkanti, multithread. Esempi di client TCP e UDP via socket per il servizio *daytime*.

Monitoraggio delle socket con il comando *netstat(8)*.

I *daemon* in UNIX. I runlevel: significato, funzionamento degli script, sequenza di startup. I servizi di UNIX (protocolli a livello applicazione): la tabella */etc/services*, le porte di Unix; porte riservate e loro motivazione. Il super server *inetd*.

43 Servizi e Sistemi distribuiti (10/3)

Obiettivi dei sistemi distribuiti. Problemi di progetto: trasparenza e località, mobilità dell'utente, tolleranza ai guasti, scalabilità, sistemi su larga scala, struttura dei processi server.

Robustezza di un sistema distribuito: rilevamento dei guasti, riconfigurazione, reintegro delle parti dopo la riparazione.

Modelli di sistemi distribuiti basati su migrazione dei dati: WWW, Lotus Notes, AFS, Coda, NFS, SMB.

44 RPC, NFS (11/3)

Modelli di sistemi distribuiti basati sulla migrazione delle computazioni (RPC, RMI, CORBA); sulla migrazione dei processi/thread (MOSIX, DCE); sulla coordinazione distribuita (Linda, Jini).

Esempi di servizi distribuiti. Schema di chiamata remota (RPC). Porta, servizio, associazione statica e dinamica (portmapper).

Il Network File System: concetti base, schema di funzionamento via RPC. Caratteristiche e architettura del sistema. Condivisione di un file system via rete: esportazione e montaggio NFS.

Riferimenti bibliografici

- [1] Andrew S. Tanenbaum. *I moderni sistemi operativi*, II edizione. Jackson, 2002.
- [2] A. Silberschatz, P.B. Galvin, G. Gagne, *Sistemi operativi* sesta ed., Addison-Wesley, 2002. ISBN 88-7192-140-2.