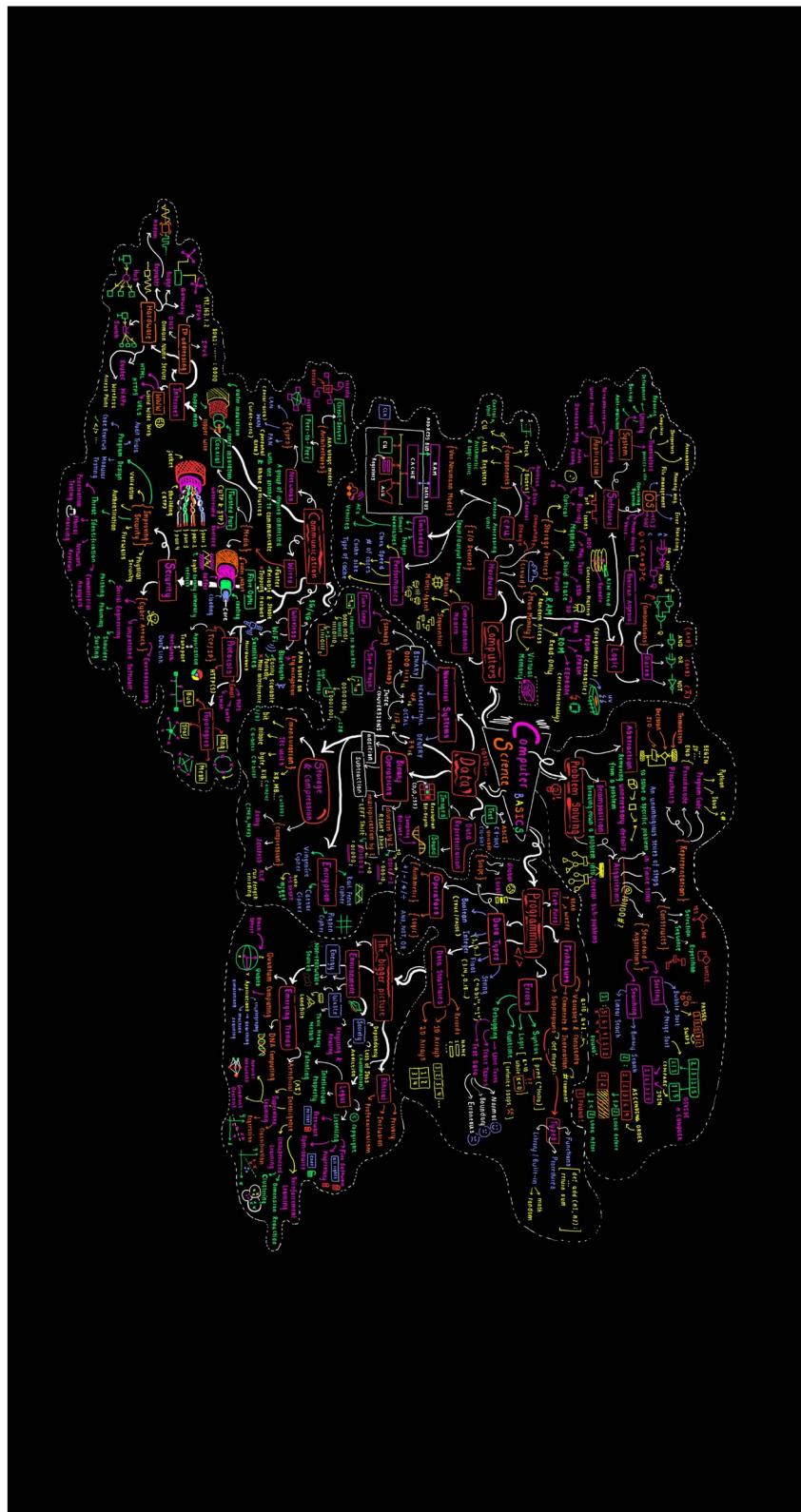


OPOSICIONES DE SECUNDARIA

TEMAS DE INFORMÁTICA

Por: Juan Andrés Morales Cordovilla
Web: <http://www.ugr.es/~jamc/>



Índice

1 Común Otras Disciplinas	5
1.1 Unidades Didácticas. Historia. Empresas. Bibliografía	5
1.2 Informática	6
1.3 Matemáticas	8
1.3.1 Aritmética, Geometría, Álgebra y Análisis	8
1.3.2 Grafos	9
1.3.3 Señales, probabilidad y estadística	9
1.3.4 Redes: peticiones, colas y flujo	10
1.3.5 Teoría de Complejidad y Métricas	11
1.3.6 Teoría de Conjuntos	12
1.3.7 Problemas y heurísticas	13
1.3.8 Correspondencia Curry-Howard-Lambek	14
1.3.9 Lógica formal	15
1.4 Otras disciplinas	18
1.4.1 Procesamiento de señales	18
1.4.2 Física	18
1.4.3 Biología	19
1.4.4 Economía	19
1.4.5 Filosofía	20
1.5 Impacto de las (nuevas) tecnologías	20
1.6 T0Otros	22
1.7 T0Otros2	23
2 Común Arquitectura	25
3 +*1. Representación y comunicación de la información. (27DicMarcos)	27
3.1 Introducción. Conclusión	27
3.2 Datos, información y conocimiento	27
3.3 Digitalización	28
3.4 Teoría de la Información	28
3.5 Codificación	29
3.6 Representación de números, texto y multimedia	29
3.7 Comunicación y tipos	29
3.8 Elementos de un sistema de comunicación	30
3.9 Problemas en la transmisión	30
3.10 Manejo de errores en comunicación	30
3.11 T1Informac	32

4 +2. Elementos funcionales de un ordenador digital. *Arquitectura (22SepJavier)	33
4.1 Introducción. Conclusión	33
4.2 Ordenador digital vs analógico	33
4.3 Historia de generaciones	34
4.4 CPU	34
4.5 Memoria	34
4.6 Periféricos	34
4.7 Bus, placa madre	35
4.8 Paralelismo a nivel de nº de procesadores: Taxonomía Flynn y memoria compartida	35
4.8.1 Memoria compartida: problema de la coherencia	35
4.9 Tipos según rendimiento	36
4.10 T2OrdDig	37
5 +3. Componentes, estructura y funcionamiento de la Unidad Central de Proceso. (5NovMigue)	39
5.1 Introducción. Conclusión	39
5.2 Arquitecturas	39
5.3 CPU. Características	40
5.4 Bus del Sistema. ALU. Caché	40
5.5 Unidad de Control	40
5.6 CODE2	40
5.7 Funcionamiento CPU	41
5.8 Juego de instrucciones reducido, complejo y extendido	42
5.9 Paralelismo Hardware y cache	42
5.10 Ciclo máquina	43
5.11 Paralelismo a nivel de instrucción: pipeline, superescalar	43
5.12 Paralelismo en nº proces: Flynn y memoria compartida	44
5.13 Microprocesadores populares	44
5.14 T3CPU	45
6 *+4. Memoria interna. Tipos. Direccionamiento. Características y funciones. (24NovCarmen)	47
6.1 Introducción. Conclusión	47
6.2 Arquitectura. Harvard modificada	47
6.3 Memoria. Jerarquia	47
6.4 Estructura de una memoria	48
6.5 Memoria: características	48
6.6 Registros	49
6.7 Caché	49
6.8 Memoria principal. RAM	50
6.9 DDR (SDRAM)	50
6.10 VRAM vs GDDR RAM	51
6.11 Conectores RAM	51
6.12 Memoria principal: ROM	52
6.13 Direccionamiento: modos de acceso a SRAM, DRAM y CAM	52
6.14 Direccionamiento: modos	53
6.15 T4MemIntDir	54

7	5. Microprocesadores. Estructura. Tipos. Comunicación con el exterior.	55
7.1	Introducción. Conclusión	55
7.2	Arquitectura del ordenador	55
7.3	Estructura estandar de CPU	55
7.4	Microprocesador: características	55
7.5	Características Físicas	56
7.6	Características Softw.	56
7.7	Comunicación con exterior: I/O	57
7.8	Comunicación con exterior: bus del sistema	57
7.9	Historia de los Micros	58
7.10	Fabricantes Actuales	58
7.10.1	Benchmark de micro actual	59
7.11	Cuantica	59
7.12	T5Micopr	61
8	*6. Sistemas de almacenamiento externo. Tipos. Características y funcionamiento.	63
8.1	Introducción. Conclusión	63
8.2	Computadora von Neumann. Jerarquia de memorias	63
8.3	Almacenamiento externo y Memoria secundaria y	63
8.4	Soportes magnéticos	64
8.5	Ópticos y magneto-ópticos	65
8.6	Soportes electrónicos o flash	66
8.7	Conectores	67
8.8	Almacenamiento distibuido. RAID. Nube	67
8.9	Otros	68
8.10	T6AImExtMS	69
9	*+7. Dispositivos periféricos de entrada/salida. Características y funcionamiento. (21EneEva)	71
9.1	Introducción. Conclusión	71
9.2	Arquitectura de computadoras. Placa madre	71
9.3	Periférico. Estructura	71
9.4	Estructura hardware de los periféricos	72
9.5	Tipos	72
9.6	Direccionamiento de periféricos: PIO	73
9.7	Puertos o conectores	74
9.8	Dispositivos de entrada: características y funcionamiento	75
9.9	Dispositivos de salida: características y funcionamiento	76
9.10	Dispositivos de E/S	78
9.11	Dispositivos de almacenamiento masivo	78
9.12	T7PerifES	79
10	8. Hardware comercial de un ordenador. Placa base. Tarjetas controladoras de dispositivos y de entrada/salida	81
11	+*9. Lógica de circuitos. Circuitos combinacionales y secuenciales. (17SepJavier)	85
11.1	Introducción. Conclusión	85
11.2	Circuitos digitales	85
11.3	Operadores lógicos	85
11.4	Álgebra de Boole: axiomas	86

11.5 Circuito combinacional. Karnaugh	86
11.6 Ejemplos de combinacionales	87
11.7 Circuito secuencial	87
11.8 Ejemplos de circuitos secuenciales	88
11.9 Autómatas programables	89
11.10 Recursos, simuladores e IDE	89
11.11 T9LogCirc	90
12 Común SO	91
12.1 20. Explotación y Administración de sistemas operativos monousuario y multiusuario.	92
12.1.1 Administración y configuración de SO	92
12.2 22. Planificación y explotación de sistemas Informáticos. Configuración. Condiciones de instalación. Medidas de seguridad. Procedimientos de uso.	94
13 +*10. Representación interna de los datos. (27DicMarcos)	97
13.1 Introducción	97
13.2 Jerarquía datos, información, conocimiento	97
13.3 Digitalización: ventajas de Shannon	97
13.4 Codificación	98
13.5 Representación de números	98
13.6 Representación de texto	100
13.7 Representación de estructuras de datos	101
13.8 Representación multimedia	101
13.9 Compresión	102
13.10 Manejo de errores	103
13.11 T10ReprIntDat	104
14 *12. Organización lógica de los datos. Estructuras dinámicas.	105
14.1 Introducción. Conclusión	105
14.2 Organización lógica de los datos: Tipo de dato abstracto vs estructura de datos .	105
14.3 Clasificación de las estructuras de datos: dinámicas no/lineales	106
14.4 Listas	107
14.5 Listas especiales: pila, cola y diccionario	108
14.6 Árbol binario	108
14.7 Árbol binario de búsqueda	109
14.8 Árbol binario de búsqueda autobalanceado	109
14.9 Grafos	110
14.10 Tabla Hash	111
14.11 T12EstrDin	112
15 *13. Ficheros. Tipos. Características. Organizaciones.	113
15.1 Introducción. Conclusión.	113
15.2 Ficheros o archivos	113
15.3 Características	113
15.4 Los 7 tipos de ficheros	114
15.5 Tipos de ficheros	114
15.6 Tipos: Sistemas de archivos no y orientados a registros	114
15.7 Sistemas de archivos no orientados a registros. Gestión de asignación y espacio libre	115

15.8 Sistemas de archivos orientados a registros	115
15.9 Organización de ficheros o métodos de acceso	115
15.10Organización secuencial	116
15.11Organización aleatoria o directa	116
15.12Organización secuencial en cadena. Blockchain	116
15.13Organización indexada: SAM	117
15.14T13FichOrgan	118
16 *14. Utilización de ficheros según su organización.	119
16.1 Métricas de uso de ficheros	119
16.2 Organización de ficheros o métodos de acceso	119
16.3 Índices, factor de bloque y búsquedas masivas.	119
16.4 Organización indexada: tipos (SAM)	120
16.5 ISAM	120
16.6 VSAM	121
16.7 Otros detalles de ISAM	121
16.8 Otros detalles de VSAM	122
16.9 T14FichOrganUso	123
17 +15. Sistemas Operativos. Componentes. Estructuras. Funciones. Tipos (9FebAn-gela)	125
17.1 Introducción. Conclusión	125
17.2 Arquitectura, recursos y cargas	125
17.3 Sistemas Operativos y funciones	126
17.4 Tipos 1: lotes, multi(tarea-usr-proc)	126
17.5 Tipos 2: RT, Red	127
17.6 Componentes: arquitectura de sistemas informáticos	128
17.7 Componentes: Arquitectura de los SO	128
17.7.1 Shell: GUI y CLI	128
17.7.2 Servicios o Daemon	129
17.7.3 Kernel e IPC	129
17.8 Arquitectura según Kernel	130
17.9 Evolución histórica	130
17.10SO actuales	131
17.11T15SOComponentes	133
18 +*16. Sistemas operativos: Gestión de procesos. (20OctEncarni)	135
18.1 Introducción. Conclusión	135
18.2 Sistema Operativo y multitarea	135
18.3 Procesos y PCB	135
18.4 Fork, Hilo (hebra) y multihilo	136
18.5 Gestión de procesos. IPC	136
18.6 Planificación de procesos. Algoritmos	137
18.7 Análisis de planificabilidad en tiempo real	138
18.8 Concurrencia	138
18.9 Problemas de la concurrencia: Sección crítica y condiciones de carrera	139
18.10Soluciones: Exclusión Mutua. Espera Activa.	139
18.11Comunicación: memoria compartida (semáforos, monitores)	140
18.12Comunicación: paso de mensajes	140
18.13Soluciones basadas en Exclusión Mutua	141

18.14	Problemas de los semáforos: Interbloqueo e inanición	141
18.15	Gestión procesos en SO populares	142
18.16	T16GestProc	143
19	*17. Sistemas operativos: Gestión de memoria.	145
19.1	Introducción. Conclusión	145
19.2	Fundamentos: sistema operativo (Kernel), jerarquía de memoria y multiprogramación	145
19.3	Hardware. Controlador de memoria. MMU	146
19.4	Gestión sin virtualizar (asignación): tabla y métodos	146
19.4.1	Asignación (gestión sin virtualizar): Fragmentación	147
19.5	Memoria virtual	148
19.6	Memoria virtual: paginación	148
19.7	Memoria virtual: segmentación	149
19.8	Paginación vs segmentación	150
19.9	T17SOMem	151
20	*+18. Sistemas operativos: Gestión de entradas/salidas. (18MarJoAnt)	153
20.1	Introducción. Conclusión	153
20.2	Sistema Operativo	153
20.3	Hardware de E/S: periféricos	153
20.4	Controlador vs driver	153
20.5	Gestión de E/S	154
20.6	Mecanismos básicos de gestión de E/S: spooling y chaching	155
20.7	Técnicas de gestión de E/S: PIO (programa), IIO (interrupción) y DMA	155
20.8	Mejoras a DMA: IOP (procesador) y canal de E/S	155
20.9	Planificación de disco HDD	156
20.10	Planificación de unidad SSD	157
20.11	Planificación de procesos con E/S	157
20.12	Gestión E/S en Linux y Windows	157
20.13	T18SOGestES	159
21	*19. Sistemas operativos: Gestión de archivos y dispositivos.	161
21.1	Introducción. Conclusión	161
21.2	Sistema operativo	161
21.3	Fichero y directorio	161
21.4	Sistema de Archivos (SA)	162
21.5	Características de los SA: Integridad, permisos y redundancia	162
21.6	Gestión del espacio: asignación y espacio libre (cuota de disco)	163
21.7	POSIX: INode y enlaces	163
21.8	POSIX: Descriptor de archivo	164
21.9	Ejemplos SA	165
21.9.1	SA Windows	165
21.9.2	SA Unix	165
21.9.3	Otros SA	166
21.9.4	SA virtual y en red: NFS y VFS	166
21.10	Gestión de dispositivos	166
21.11	POSIX: Archivos de dispositivos: orientados a bloque o carácter	166
21.12	Planificación de disco: HDD y SSD	167
21.13	T19SOGestArchDisp	168

22 21. Sistemas informáticos. Estructura física y funcional.	169
22.0.1 Computador. Sistema Turing completo	169
22.0.2 Informática. Wares	169
22.0.3 Alcance y potencia de sistemas	169
22.0.4 DSP y embebidos	171
23 Común Alg	173
23.1 29. Utilidades para el desarrollo y prueba de programas. Compiladores. Intérpretes. Depuradores.	174
23.2 30. Prueba y documentación de programas. Técnicas.	175
23.3 33. Programación en lenguaje ensamblador. Instrucciones básicas. Formatos. Direccionamientos.	175
24 +23. Diseño de algoritmos. Técnicas descriptivas. (27OctJoseA)	177
24.1 Introducción. Conclusión	177
24.2 Algoritmo vs programa	177
24.3 Ciclo de desarrollo de software	178
24.4 Diseño de algoritmos: estructural, objetos e interfaces	178
24.5 Técnicas descriptivas de diseño	180
24.6 Diseño: técnicas para eficiencia	180
24.7 Paradigmas de diseño de algoritmos	181
24.7.1 Ordenación y Busquedad	181
24.7.2 Paradigmas de optimización	182
24.8 T23DisAlg	184
25 +24. Lenguajes de programación. Tipos. Características. 15OctAngela	185
25.1 Introducción. Conclusión	185
25.2 Lenguaje de programación. Chomsky	185
25.2.1 Según cercanía a máquina	185
25.3 Segundo generación (o paradigma)	186
25.3.1 Paradigmas de Programación	186
25.4 Otras clasificaciones de los lenguajes	188
25.4.1 Según compilación	188
25.4.2 Según concurrencia	188
25.4.3 Según tipo de aplicación u objetivo	188
25.5 Componentes comunes de los lenguajes	188
25.6 Curva de aprendizaje y popularidad	189
25.7 Lenguajes populares por orden cronológico	189
25.8 T24LengTipos	191
26 +25. Programación estructurada. Estructuras básicas. Funciones y Procedimientos. (4MarPedro)	193
26.1 Introducción. Conclusión	193
26.2 Fundamentos	193
26.3 Programación estructurada y teorema	193
26.4 Diseño Top-down	194
26.5 Estructuras de control	194
26.5.1 Secuenciales	195
26.5.2 Condicionales	195
26.5.3 Iterativas	195

26.5.4 Rupturas	196
26.6 Subrutinas y Funciones	196
26.7 Parámetros, llamadas y pasos	196
26.8 Ámbito de un identificador, variable global	197
26.9 Programación en la actualidad	197
26.10T25PrEstr	198
27 26. Programación modular. Diseño de funciones. Recursividad. Librerías.	199
27.1 Introducción. Conclusión	199
27.2 Programación	199
27.3 Funciones y subrutinas	199
27.4 Programación modular y módulo	199
27.5 Ventajas y características: Transparencia, ámbito y refactorización	200
27.6 Espacio de nombres vs ámbito	200
27.7 Diseño de Funciones: axiomatización ortogonal	201
27.8 Iteratividad vs recursividad	201
27.9 Librerías	202
27.10La programación en la actualidad	203
27.11T26PrMod	204
28 *+27. Programación orientada a objetos. Objetos. Clases. Herencia. Polimorfismo. Lenguajes. (26EneElisa)	205
28.1 Introducción. Conclusión	205
28.2 Programación orientada a objetos	205
28.3 Características: encapsulamiento y paso de mensajes	206
28.4 Clases	206
28.5 Método, constructor y operador resolución de ambito	206
28.6 Objetos e instancias	207
28.7 Herencia y diagrama UML	207
28.8 Polimorfismo. Función virtual	208
28.9 Polimorfismos: Sobrecarga de operadores	209
28.10Lenguajes orientados a objetos	209
28.11La programación en la actualidad	209
28.12T27POO	211
29 *+28. Programación en tiempo real. Interrupciones. Sincronización y comunicación entre tareas. Lenguajes. (17DicFrancis)	213
29.1 Introducción	213
29.2 Definición de tiempo real: computación, SO, L, sistema	213
29.3 Sistemas de RT, clasificación y aplicaciones: DSP y embebidos	214
29.4 Interrupción	214
29.5 Concurrencia	215
29.6 Sistemas operativos de tiempo real	215
29.7 SO: Planificabilidad en tiempo real	215
29.8 SO: Herramientas de diseño y análisis	216
29.9 Programación: estándares	216
29.10Programación: concurrente fork y wait	217
29.11Programación concurrente: ejemplos en C y Java	217
29.12Programación: orientada a eventos	218
29.13T28TRInterrSincr	219

30 +31. Lenguaje C: Características generales. Elementos del lenguaje. Estructura de un programa. Funciones de librería y usuario. Entorno de compilación. Herramientas para la elaboración y depuración de programas en lenguaje C. (9Mar-Fuen)	221
30.1 Introducción. Conclusión. Lenguaje C	221
30.2 Características generales	221
30.3 Elementos tokens	222
30.4 Elementos: descriptores de datos: tipos, variables y struct	223
30.5 Elementos: descripción de acciones: expresiones y estructuras de control	223
30.6 Estructura de un programa	223
30.6.1 Preprocesado, directivas y macros	224
30.7 Funciones de librería y usuario	224
30.8 Entorno de compilación	224
30.9 Herramientas para la elaboración y depuración	225
30.10T31CGeneral	226
31 *32. Lenguaje C: Manipulación de estructuras de datos dinámicas y estáticas.	
Entrada y salida de datos. Gestión de punteros. Punteros a funciones.	227
31.1 Introducción. Conclusión	227
31.2 Lenguaje C	227
31.3 Entradas salidas	227
31.4 E/S mediante ficheros	228
31.5 Manipulación de estructuras estáticas	228
31.6 Punteros	230
31.7 Asignación dinámica de memoria	231
31.8 Arrays, Listas,..	231
31.9 Gráficos y videojuegos en C (solo SAI)	232
31.10T32CEstrDinEstIO	233
32 Común BD	235
32.1 Persistencia de objetos (T29Borrador)	235
32.2 35. La definición de datos. Niveles de descripción. Lenguajes. Diccionario de datos.	236
32.3 36. La manipulación de datos. Operaciones. Lenguajes. Optimización de consultas.	236
32.4 37. Modelo de datos jerárquico y en red. Estructuras. Operaciones.	236
32.5 40. Diseño de bases de datos relacionales.	237
32.6 42. Sistemas de base de datos distribuidos.	237
32.7 44. Técnicas y procedimientos para la seguridad de los datos.	239
33 +34. Sistemas gestores de base de datos. Funciones. Componentes. Arquitecturas de referencia y operacionales. Tipos de sistemas. (6OctJoaq)	241
33.1 Introducción. Conclusión	241
33.2 Base de Datos	241
33.3 Sistema Gestor de Bases de Datos y funciones	242
33.4 Tipos de modelos de BD	242
33.5 Arquitectura de referencia	242
33.6 Arquitecturas operacionales	243
33.7 Componentes de un SGBD	243
33.8 Herramientas de desarrollo de aplicaciones	244

33.9 SGBD Populares	244
33.10 MySQL, MariaDB y PhpMyAdmin	244
33.11 T34SGBD	246
34 +38. Modelo de datos relacional. Estructuras. Operaciones. Algebra relacional.	
(10NovFuen)	247
34.1 Introducción. Conclusión	247
34.2 Base de datos y fundamentos	247
34.3 Modelo Relacional	247
34.4 Estructura: el esquema	248
34.5 Elementos básicos de una relación o tabla	248
34.6 Claves y relación de tablas	248
34.7 Normalización de tablas	249
34.8 Restricciones a los datos	249
34.9 Teorema de Codd. Declarativo vs procedural	250
34.10 Calculo relacional	250
34.11 Operaciones y lenguajes del cálculo relacional	251
34.12 Algebra relacional	251
34.13 Operaciones del álgebra relacional	251
34.14 T38AlgRel	253
35 +*39. Lenguajes para la definición y manipulación de datos en sistemas de base de datos relacionales. Tipos. Características. Lenguaje SQL. (18FebJoaq)	255
35.1 Introducción. Conclusión	255
35.2 Bases de datos relacionales	255
35.3 Lenguaje de definición de datos	255
35.4 Lenguaje de control de datos	256
35.5 Lenguaje de manipulación de datos. Teorema de Codd	256
35.5.1 Lenguajes comerciales (aparte de SQL): QUEL, QBE y FQL	256
35.6 SQL	257
35.6.1 DDL: CREATE	257
35.6.2 DDL: ALTER/TRUNCATE/DROP TABLE	258
35.6.3 DML: INSERT INTO, DELETE FROM, UPDATE	259
35.6.4 DDL???: VIEW, CREATE INDEX	259
35.7 DQL: SELECT	259
35.7.1 DQL???: SELECT INTO, AVG-GROUP, COUNT	261
35.7.2 Procedimientos de Almacenamiento: PROCEDURE	261
35.8 T39SQL	262
36 41. Utilidades de los sistemas gestores de base de datos para el desarrollo de aplicaciones. Tipos. Características.	263
37 Común Ingeniería del Software	265
37.0.1 Arquitectura del software	265
37.0.2 Despliegue	266
37.0.3 Generalización y patrones	266
37.0.4 DevOps y disponibilidad	267
37.1 54. Diseño de interfaces de usuario: Criterios de diseño. Descripción de interfaces. Documentación. Herramientas para la construcción de interfaces.	268
37.2 56. Análisis y diseño orientado a objetos	268

37.3	57. Calidad del «software». Factores y métricas. Estrategias de prueba.	268
37.3.1	Control de versiones y calidad	268
37.3.2	Factores y métricas.	269
37.3.3	Estrategias de prueba.	269
37.4	59. Gestión y control de proyectos Informáticos. Estimación de recursos. Planificación temporal y organizativa. Seguimiento.	270
37.4.1	Gestión y control de proyectos Informáticos. Estimación de recursos.	270
37.4.2	Planificación temporal y organizativa. Seguimiento.	271
37.5	58. Ayudas automatizadas para el desarrollo de «software» (herramientas CASE). Tipos. Estructura. Prestaciones.	271
37.6	T0Ing	272
38	60. Sistemas basados en el conocimiento. Representación del conocimiento. Componentes y arquitectura. (un poco IA)	273
38.1	Historia	273
38.1.1	Inteligencia Artificial y Bioinformática	273
39	Común Redes	277
39.1	73. Evaluación y mejora de prestaciones, en un sistema en red. Técnicas y procedimientos de medidas.	282
39.2	74. Sistemas multimedia	282
40	*+61. Redes y servicios de comunicaciones. (1OctMarcos)	285
40.1	Introducción. Conclusión	285
40.2	Red	285
40.3	Clasificación de las redes	286
40.4	Arquitectura por capas y componentes básicos de redes	287
40.5	Servidor y host	287
40.6	Servicios	288
40.7	MPLS	289
40.8	Otras superadas por MPLS	289
40.9	Ejemplos de redes	290
40.10	T61RedServic	291
41	*+62. Arquitecturas de sistemas de comunicaciones. Arquitecturas basadas en niveles. Estándares. (19NovLola)	293
41.1	Introducción. Conclusión	293
41.2	Sistemas de comunicaciones. Servicios y servidores.	293
41.3	Computación o aplicación distribuida	293
41.4	Pila de protocolos	294
41.5	Paquetes y cabeceras	295
41.6	Modelo OSI	296
41.7	Protocolo TCP/IP	298
41.8	Comparación OSI con TCP/IP	298
41.9	Organizaciones de estándares	299
41.10	T62ArqComNiv	301

42 +*63. Funciones y servicios del nivel físico. Tipos y medios de transmisión. Adaptación al medio de transmisión. Limitaciones a la transmisión. Estándares. (8DicJuanA)	303
42.1 Introducción. Conclusión	303
42.2 Sistemas en Red	304
42.3 Funciones y servicios del nivel físico OSI	304
42.4 Fundamentos: Nyquist, Fourier, ancho de banda, canal, Shannon-Hartley	304
42.5 Comunicación. Transmisión. Medio de Transmisión	305
42.6 Medios de transmisión guiados: los 3 tipos	306
42.7 Medios de transmisión: no guiados (Wifi)	308
42.8 Técnicas de adaptación: codificación y modulación	309
42.9 Limitaciones a la transmisión	310
42.10 Otras	311
42.11 Estándarización de las redes	312
42.12 T63MedTransm	313
43 64. Funciones y servicios del nivel de enlace. Técnicas. Protocolos.	315
43.1 T64NEnl	317
44 65. Funciones y servicios del nivel de red y del nivel de transporte. Técnicas. Protocolos.	319
44.1 Nivel de Red	319
44.1.1 Protocolos de encaminamiento	321
44.2 Nivel de transporte	322
44.3 T65NRedNTransp	325
45 66. Funciones y servicios en niveles sesión, presentación y aplicación. Protocolos. Estándares.	327
45.1 Sesión	327
45.2 Presentación	327
45.3 Aplicación	328
46 *+67. Redes de área local. Componentes. Topologías. Estándares. Protocolos. (3DicMariCarmen)	329
46.1 Introducción. Conclusión	329
46.2 Arquitectura de redes	329
46.3 Red de Área Local	329
46.4 Topología: red bus	330
46.5 Componentes: clasificación	331
46.5.1 Componentes: Switch	331
46.5.2 Componentes: gateway residencial (modem), WAP y tarjeta	331
46.5.3 Componentes de instalación: par trenzado y racks	331
46.6 Componentes Software	332
46.7 Estándares: IEEE 802	332
46.8 Ethernet	332
46.9 Wifi	333
46.10 Colisiones: VLAN y circuitos en Ethernet	334
46.11 Protocolos de acceso a Internet	334
46.12 Protocolos antiguos	334
46.13 T67LANCompProt	335

47 +68. Software de sistemas en red. Componentes. Funciones. Estructura. (4FebLola)	337
47.1 Introducción. Conclusión	337
47.2 Sistemas en red	337
47.3 Software de red. Clasificación. Importancia de los SO	337
47.4 Softw. red: funciones principales	338
47.5 Sistemas operativos de red	338
47.6 SO ligeros: Stack solucion	339
47.7 Sistemas operativos para servidores: estructura	339
47.8 Servicio de directorio: estructura jerárquica	340
47.9 Almacenamiento Distribuido	341
47.10 Softw. red: monitorización y gestión	342
47.11 Softw. red: aplic.	342
47.12 Softw. red: pila de protocolos	342
47.13 Kubernetes	342
47.14 Otros: remoto, teletrabajo	342
47.15 T68SoftwRed	343
48 *+69. Integración de sistemas. Medios de Interconexión. Estándares. Protocolos de acceso a redes de área extensa (7EneCarmenCampos)	345
48.1 Introducción. Conclusión	345
48.2 Arquitectura TCP/IP y protocolos	345
48.3 Integración de sistemas en red	345
48.4 Proveedor, tier1 y PDN	346
48.5 Hardware de red	346
48.6 Medios de interconexión en capa física	347
48.6.1 En capa de enlace	347
48.6.2 En capa de red: routers y gateway residencial	347
48.6.3 En capa de transporte y aplicación	348
48.7 Resumen de protocolos por: acceso a Internet desde LAN	349
48.8 Protocolos LAN de acceso a WAN y de integración: PPP	349
48.9 Protocolos en capa de acceso	349
48.10 Protocolos: En capa de red: VLAN y STP	350
48.11 Protocolos de integración	350
48.12 Protocolos en Capa aplicación	350
48.13 Ejemplos de LAN a WAN	350
48.14 T69IntegrProtWAN	351
49 +70. Diseño de sistemas en red local. Parámetros de diseño. Instalación y configuración de sistemas en red local. (12EneAlex)	353
49.1 Introducción. Conclusión	353
49.2 LAN	353
49.3 Diseño de sistemas en red local.	353
49.4 Parámetros de diseño.	354
49.5 Instalación de LAN	354
49.6 Instalación: centro datos	355
49.7 Configuración de LAN: componentes	356
49.8 Configuración de LAN: enruteamientos	357
49.9 Sistemas en red local	358
49.10 T70DisInstLAN	359

50 71. Explotación y administración de sistemas en red local. Facilidades de gestión.	361
50.1 Introducción. Conclusión	361
50.2 LAN	361
50.3 Explotación de LAN	361
50.4 Administración	362
50.5 Facilidades de gestión.	363
50.6 T71AdminLAN	364
51 +72. La seguridad en sistemas en red: Servicios de seguridad. Técnicas y sistemas de protección. Estándares. (23FebEncarni)	365
51.1 Introducción	365
51.2 Ciberseguridad	365
51.2.1 CIDA (no definidos en otros sitios)	366
51.3 Funciones unidireccionales	367
51.4 Criptografía simétrica y asimétrica	367
51.4.1 Ej. criptografía simétrica: WPA	368
51.4.2 Ej. criptografía híbrida: TLS	368
51.5 Autenticación	369
51.5.1 Certificado vs firma digital	369
51.6 Ataques	370
51.7 Malware	370
51.8 Suplantación e ingeniería social	371
51.9 Vulnerabilidades: resumen de las mencionadas	371
51.10 Protección: cebolla	371
51.11 Seguridad perimetral: interceptación	372
51.12 Seguridad perimetral: intrusiones	372
51.13 Seguridad perimetral: firewall y proxy	373
51.14 Seguridad perimetral: VPN y SSH	373
51.15 Estandares: resumen de los mencionados	374
51.16 Ley LOPDGDD	374
51.17 Fakenews y ciberacoso	375
51.18 T72SegRed	376
52 Fórmulas en problemas	377
52.1 Tablas	377

Chapter 1

Común Otras Disciplinas

1.1 Unidades Didácticas. Historia. Empresas. Bibliografía

- *UIArquit ... Andalucia*
- **IMPORTANTE:** revisar que las secciones cumplen el título.
- *B.* Bibliografia **IMPORTANTE** buscar en Hist. y rellenar
 - Hist.: [Garden] (8 Inteligencias, *Shannon*), [Goldstine] The Computer from Pascal to von Neumann. Princeton UniversityPress, 1980. *Neumann, Shannon*) [Chaitin] (*TAI*).
 - A. [Lloris00] DiseñoLog [Prieto01] Alberto P. Introducción a la Informática. (2006) [Penrose] (*Turing*)
 - SO
 -) [Tanenbaum] Andrew S. T. Sistemas operativos modernos. 1992 (2007)
 -) [Stallings] William S. Sistemas Operativos. Aspectos Internos Y Principios De Diseño. 1992 (2005)
 -) [Starling]??
 - L. [Knuth] Donald K. The Art of Computer Programming. 2010. *BigO*
 - BD: [Date04] ver *Rel. Tab.*
 - R. [Stallings] William S. Comunicaciones y redes de computadores. 1997. (*TCP/IP*) [Feynman] Richard F. Conferencias sobre computación. 1996.
 - Webs:
geeksforgeeks.org (SO),
stackoverflow.com,
w3schools.com (BD, Web),
tiobe.com (L),
- *Hist. A. Hist. SO, Hist. L., Hist. BD , Hist. I. , Hist. R. ,*
- *Empresa* , Organización, Universidad o Centro *Tecn..:*
 - *Intel* (ver)
 - *Oracle* Corporation: por Ellison70 para su *SGBD*. En 2010 compró Sun Microsystem (*Solaris, Java, NFS (Almac. Distrib.)*). Destaca: Oracle DB, VirtualBox (*Maq. Virt.*). Cuida marca *JavaScript*.

- **Bell** Lab.: antes era de ATT (fundada por Alexander Bell en 1880) ahora de Nokia (destaca *Tier1*). Destacan novedades e inventos: *Transistor*, Fernando Corbató (Multics Hist. Linux), Ken **Thompson** (*Turing B y Unix*, *Unicode UTF – 8*, *Comando grep*, usó *DEC PDP-11*) *C*, *Par Trenz.*, *ADSL*, *Shannon* (T. *Inform.*), *Muestr.* (Nyquist)
- **IBM, DEC**
- TAP: *CISCO (Red)*, *Kingston* (Mem.), *Intel* (CPU)
- **HP** (Hewlett Packard): *Supercomput.* Cray, *UniBus (MIO)*
- **DARPA** (Defense Advanced Research Projects of USA, *Milit.*): ARPANET (*TCP/IP*), *COBOL*, GPS (*Satelite*), *L. ADA*
- *CISCO*, Novell (*IPX/SPX y Netware*).
- Netscape (no confundir con *Netware*=*SO Servid.*): creó *JavaScript* y *Mozilla* (Guerra Naveg.) y SSL (*TLS*)
- E. de IA. Nvidia (*GPU*, Stable Diffusion VA (ver abajo))
- Gigantes Tecn.: **Google** (Colab (*Nube*), Naveg. Chrome, *Android*, *AJAX*), **Amazon** (ver *NoSQL*, AWS *BD Nube*, Stable Diffusion VA), Microsoft y Facebook. G. Home (*IoT*)
- Microsoft: *Win*, *Videojuego*-consola *Xbox*
- Univ.: **MIT** (*LISP*, *Shannon*), Berkley: *QUEL* Ingress, *FreeBDS*, *RISC-V*, *SSH* (*rsh*), *Kerberos* (*Cript.*) *Libre*: *GNU (GCC,...)*

Ver *Sucursal*, *Popular*, PYME (Pequeña Y Mediana E.), *Proveedor*, *WPA E.*

1.2 Informática

- **Unif.** : ver *Filos.*

- Hardw.: *Mem.*, *Proc.*, *Coher.* (*Actual.* *Cache*), Genialidades de *Boole* y *Shannon* (*Digit.*), *Acces.* *Aleat.*, *Compo.* *Opt.* (*Conect.*). *Superescalar* y *DDR RAM* (mantienen *Moore*).
- *SO*: *Recurso T/S*, *Interfer.*, *Concurr.*, *Cola-Petic.-Buffer-Trim-Basura*, *Prod.*, *Esper.* *Activa+Interr.*, *I/O*, *Grup.* vs *Jerarq.*, *Lista Enl.=Punt.=Dinam.=Blockchain*, *4 Tablas SO* (todo *Fich.*). *Mem.* *Virt.*, *Fich.* *Unix*.
- *BD y Dat.*: *Estr.* *Dat.=Form.=Estr.* (ver abajo *Graf.*), *Consult.=Topos*, *Vista*, *Consult.* o *Busq.* *Vel.* (*Ind.*)
- *Alg.*: *L.*, *Impl.*, *Recurs.*, *Virt.*, *Estr.* *Control*, *Diagr.*, *Condic.*, *Obj.+Recurs.*, *Correspond.* *CHL (Cat.)*. *Equiv.* por *Exten.* o *Reduc.* (ej: *Cl. Struct*, *Her.*), *Modul.* (*Pila Protoc.*)
- *Ing. Softw.*: *Eleg.*, *Capa*, *Arq. Softw.*, *Compo.*, *Div.* *Venc.*, *Gener.*, *Axiom.* *atización Ortog.* (*Cohes.*).
- *Red*: *Interop.*, *Transpar.=Tele*, *Naveg.*, *Red Superp.* (*Medio Transm.* *Lineal*). *Mashup*, *Red Bus* (*Colis.* *Recurso*), *QoS*, *Fulkerson* y *Flujo*, *OrCo* (3 pasos, *Socket*). *Pila Protoc.* (*Modul.*) *E2E* y *AcRe* (*Internet TCP* vs *IP*)
- Todos: *Err.*, *Redund.*, *Recurso*, *Topos* (*Id.-Instanc.=Busq.* *Binar.=Naveg.*, *Dir.*, *Direc...*), *Graf.=*(*Estr.**Diagr.*, *Topol.*, *Esquema*, *Descr...*), *Informat.* (*TAP* y *Vel.*), *Map.* *Vect.* (por *Texto-Web-HTML*). *Coher.* (*Consist.* en *Compo.* para *Cooper.*), *Compo.*, *Mem.* *Compart.*, *QoS=RT*, *Kubernetes* (*SO+Red*), *SQL InYec.* (*BD+Red*)

A SO	x	SO M. Virt. x	L. Chomsky C-Unix, API- Bibl.,Concurr.	BD FactBlock EM- Transac. Precomp., B+, TmaCodd x	Ing.	R. IoT Kubernetes
L BD			x		Modul.	Php SQLInyec. PhpMyAd- min, mysqli Dis. Topol. x
Ing. R.					x	

Input	Alg. F.	Output	Pin I/O, Traduc. (Compil.)			
Abstr. Alg.			L.			
R (Read)	Mem.	Descr. Cod. Progr.	Pin R/W			
Direc.=Nombre	Mem./Busq./	W (Write)				
I (Insert.)	BD, Estr. Dat.	Dat=ValorVar	SGBD I/D			
T (Transm.)	Canal	D (Delet.)				
Sens.	Alg.	R (Recibir)	T/R Transcep.			
			Transduc.	Alg.=Ec.		
			Fisic.:	Dif.+Condic.		
			ME+ST+I	Cont. Opt.		

SO Apl.	Obj. (Tip.) de Co- munic.) IPC (Mem.) API (Mem.)	Red	Medio, (Mem.)	Socket	
------------	---	-----	------------------	--------	--

- **Sist.** : Conj. de elem. que interactuan Alg.. Los hay de dos tipos Hardw. y Softw. y poseen Arq. y Estr. Es un Unif..
 - **Sist. Abierto** : en Teor. Sist. es un tip. básico junto con S. Aislado y S. Cerrado. S. con varios I/O donde el O. es I. también (i.e. es Recurs.). Implica InterConect. (con el medio). Ver Biolog. OSI
 - **Blackbox** : forma de Conoc. un Sist. = Encapsul.: = Filos. T. Cat. (Sint. vs Anal.)=Modul. (Acces.)=Fuente Inform..

Ver Bibl.. Fuente Inform., Fisic. (=Abstr. S.), Recurso, Vulner.

- **Interop.** /Compatibilidad: capacidad de diversos Sist. de Compo. para Comunic. o Proc. (trabajar conjuntamente) con la finalidad obtener beneficio mutuo. Es similar a Cooper. u Op. o InterConect. o Integr. o Interact.

- Principio **Compo.** de *Frege* (*Log.1, Curry*): *Semant.* (significado) de *Expr.* completamente determin. por signif. de sus *Expr. Atom.* y *Estr.* (o reglas de *Combin.* o *Sintax.*). La *Subst.* es el acto de C. Es *Unif.*. Ver *Cat.*, *Interop.*, *Progr. Modul.*, *Div. Venc.*, *Binar.*, *Join*, *Sintax.*, *Atom.* *Sintax.*, *Arq.*, *Dispos.*, *Mem.*, *QUEL*, *Hot*, *Pipe*, *RT*, *IoT*, *Integr.* (*Cooper.*, *Consist.*, *Coher.*), *Progr. Compo.*, *Estr. Dat.*, *Part.*, *C. Opt. Conect.* (*Equil. Flujo*).. DesC. Fact. *Primos*
- **Transpar.** o *Encapsul.* u **Ocult.** (según “mires el vaso”): **Automágicamente** o **Sensación** o **Percepción** de no cambio (ej. *Multitar.*). Si el *Sist.* informático después de un cambio (como nueva *F.* o nuevo componente,) se adhiere a la *Interf.* externa anterior en la medida de lo posible mientras cambia su comportamiento interno. Gracias a misma *Llam.* F. Ej: *Arq. Capa*, *Multitar.*, *Concurr.* en *Recursos*, *Encapsul.* en *POO*, *Protoc.* *Red.*, *SO Distrib.*, *Arq. ANSI/SPARC*. Es *Tele* de *Red*. Es *Unif.*. Ver *Blackbox*, *Reus.*, *Circ. Virt.* (*TCP*), *Pila Protoc.*, *Servic.. Modul.* (*ReFact.*, *Ambito*), *Integr.* *Red*, *Redund.* *RAID*, *Persist.*, *VFS*

Es *Unif.* en *Biolog..* Ver *Traduc.*, *Interf.*, *ATP*, *Mashup*, *Estand.*, *Libre*, *Estand.*, *Placa Madre*, *CORBA* (*Obj.* *Distrib.* *Progr. Compo.*), *POSIX*, *Vista*, *POSIX*, *Llam.*

- **Interf.** (no confundir con *Interfer.*): *Conect. Softw.* que *Traduc.* y permite la *Interop..*
 - L.: *Llam.* *F.*, *API*, *GUI/CLI*, *Bibl.*, *Driver*
 - R.: *Gateway*, *Serie 232*, *Socket*, *Mashup*

Es un *Unif.* Ver *DMA*, *GUI*

1.3 Matemáticas

- **Matem.** : ver *Aritm.* *Geom.*, *Anal.*, *Topol.* *ATP Buzzard*, *Ley*, *Probl.*

1.3.1 Aritmética, Geometría, Álgebra y Análisis

- **Aritm.** : ver *ALU*, *Primo*
- **Geom.** : estudio de los *Graf.* *Topol.* con pesos o del *Esp..* Ej: Geodesica (*Progr. Dinam.*) y Prop. *Markov*. Ver *Conoc..*
 - **Fract.** : Ver *Vect.*, *Gener.* *Proced.* *Videojuego*, *Seis Grados*, *Compr.*, *Topol.*

Ver *3D* (*Proyec. Render.*).

- **Algebr.** (no confundir con *Alg.*): ver *Matr.*, *Rot.*, *Boole*, *Reescr.*, *Alg. Rel.*
 - **Ortog.** : dos o varios elementos lo están si son “lo más diferentes posibles”. ver *Axiom.*, *FDM*, *Acopl.-Cohes.*, *Persist.*
- **Anal.** (no confundir con *Anal. Desarr.*):
 - A. Numer.: para integrales sin definir... *C, Fortran, MATLAB*

Ver *Opt.*, *Monit.*, *Test*, *Filos.* T. *Cat.*, *Desarr.*, A. *Req.*

1.3.2 Grafos

- **Graf.** (no confundir con Gráficos-Graph 3D).: *Tupla* de dos tipos de obj.: *Nodo* (vértices, puntos) y arcos (aristas, líneas o enlaces) que *Rel.* Nod. Otra def. ver *Estr. Dat..*

– *Topol.*: en *Matem.* representada por un G. vs *Geom.*, ver *Topol. Red.*.

Es *Unif.* (=Diagr.=Topol. Red=Esquema=Descr.). Ver Sist. Comunic., Circ., Diagr., NP, Arbol, Dinam.

- **Nodo** : ver *Red, Escal., Lista Enl.*)
- **DiGraf.**: se usa en *Concurr.*
 - *Probl.* Conexión Aviones: dadas M_i de *Correspond.* no Univoca, podemos *Compo.* (multipl. M) y encontrar conexiones o no.
 - *Met. Rut. Crit.*: para *Proyec.*

Ver *Fulkerson*

- *Probl. Clique*: dado un G. buscar C. (subconj. de Vert. totalmente conectados). Es *NP*-Completo. Ver *Conoc., Cluster, Cohes.*
- **Hipergraf.** : es una *Gener.* de *Graf.* donde un arco puede conectar varios vértices a diferencia del *Graf.* donde solo conecta 2. Fig. 1.6. Similar a *MultiConj.*. Ej: *Red Bus* o *Circ. Electr.* Fong dice que la *Cat.* de H. es la EA adecuada para modelar la *InterConect.* de *Sist. Abierto*. Ver *Wolfram*
- **Jerarq.** : *Estr.* en forma de *Arbol* mas restrictiva que *Grup..* Ver *Dir., Diagr. Her., EA, Chomsky, Nivel, J. SubL. SGBD, Model. BD J. J. Bus*

1.3.3 Señales, probabilidad y estadística

- **Signal** :
 - *Proc. S.*: ver P. *Inform., Analog. o Digit., Filtr.*
 - SO: (ver *IPC POSIX*)
 - **Puls.** (Spike): ver Conformación de P. *Distors.*, Teclado (*Perif.*), *Fibra, Modulac.* PWM, *Signal Digit.*
- Ver *Sincr., Medio Transm., Modulac., Distors.. Offset, Electr., Audio, Voz, Music., DSP (RT, auricular cancelador de ruido)*
- **Filtr.** : ver *Map., Comando Grep, Imped., Tip., Switch, Firewall, Autent., Err., Signal, Ruido* (sal y pim.), *Email*
- **Prob.** y **Aleat.**: IDEA Das Leben hast impredecibles systems damit Gott hilft einbischen und wir finden nicht dass Zauber ist. z.b. Aguadulce Anrufen.
 - Mutuamente *Excluyentes*: Sucesos **Disjuntos** o ej: Dado.
 - **Markov** : un *Sist.* lo es si sus elementos solo interactúan o *Depend.* del vecino de al lado. Da un Retículo *PreOrd..* Es un tipo de *PGM*. Ver *Prob., Emerg., Cache, Progr. Dinam., LPC, M-Mills (Desarr.). Aprend. Autom. Refuerz.* (MDP)
 - *Acces. Aleat.*: ver *RAM, MS.*
 - *PseudoAleat.*: caos. ver *F. Hash, Concurr.*

Ver *Entropia, Err., Cache, Aprend. Autom., Est., F. Hash, Depend.*

- *Est.* (no confundir con Estado):
 - Auto*Corr.* : es otra forma de *Cod.* y obtener *Transf.* Fourier. Ej.: *Ruido* blanco. Ver *2FN, Depend.*
 - *PCA* : Principal Component Analysis. Ej: *MP3, JPG*
 - *PGM* : (*Prob. Graf. Model.*.) Bayesianos, *Markov, Model.* E. o *Prob.*: incluye *Alg. Prob.*, , etc. Ej. de *Aprend. Autom.=Alg.*

Ver *Efic. IDE, SGD, Monit. Red, Compr., Ruido, Visual Dat. Masiv.*

1.3.4 Redes: peticiones, colas y flujo

- *Petic.* (Request): de usar un *Recurso* que se acumulan en *Cola* o *Buffer* y un *Servid.* va atendiendo cuando pueda. Es *Unif.* (*Filos.*). Ver *IRP* (Gest. *Win Arch. Dispos.*), *Plan. Disco, Servid. Web, Servlet Applet, Proxy, RFC, Interr.*, *MultiTier, PROCEDURE (SQL)*, *CORBA* (*Obj. Distrib. Progr. Compo.*). *Spooling, DMA* (*PIO,..*), *Esper.* Activa, *Trim, Impr.* Red, *Ataq. DoS, Journal. Prod.*
- T. *Colas*: *Lista FIFO* (ver) Aplic. a *SO, Econom.*
 - *Traf.* ico Erlang: $E = \lambda h$ Metr. adim. de ocupación de una *Red*. Ej: $E = 1$ un *Recurso* (*Canal* o *Circ.*) usado siempre o 2 R. usados al 50%. Ver *Vel., Control Flujo Datos, Congest., Mejor Esfuerzo (QoS)*.
 - Disciplinas de C.: *FIFO, LIFO,...* alg. *Plan. Proc. o Disco (SCAN-Brazo)*, forma de seleccionar los miembros de la C. para recibir su *Servic.*
- *Buffer* (no confundir con *Cola*): *Mem.* intermedia para *InterConect.* dos *Proc.* que *Prod.* a distinta *Vel.* y tamaño *Bloq.*. Mejora *Efic.* del Sist. evita *Cuello Botella*. Sirve para *IPC* o *Esper.* Ej. *Perif.* usa *MP*. Ej: *Cache Disco*. Puede ocurrir *Overflow*. Es *Unif.* Ver *Spooling, fflush (C), B. Overflow (Vulner.), Arch. Dispos. Block* (con *Bu.*)
- *Overflow* : pasarse en *TAP*. Puede ocurrir en *Fich. (ISAM), Buffer, Petic., Cola* Ver *Colis., Hash.CODE2, Manej. Excep. Web stackoverflow, Tip., O. Enteros (scanf C), Buffer O. (Vulner. Segmentfault Null), Segment..*
- *Red Flujo* :
 - Alg. Ford-*Fulkerson* 1956 o *Impl. Edmonds-Karp* (21 *NP*) 1972 (*Hist. R.*): *Alg. Voraz* que resuelve en *BigO* (mn^2) el *Probl. Flujo Max.* de una *Red*: ¿F. max que puede enviar de *S* a *T*? dado un *DiGraf.* con *Nodo* fuente *S* y sumidero *T*, *Canales f/c* (Flujo varía según *Iter.* del *Alg.* y Capacidad constante o máxima) y con L. Kirchhoff $\sum_{\text{entrantes}} = \sum_{\text{salientes}}$. Usa *Busq. BFS o DFS*. IDEA: la naturaliza lo *Comput.* con **Fluidos** rápido (no *NP*). Es *Unif.* Ver *Rut. Crit.*
 - *Control F. Dat.* (Flow Control of Data): *Alg.* para prevenir *Congest.* mediante *Sincr.* y *Vel.* Transm. sostenible por Recept. desde el Transm. Ver *Diagr. F. QoS, Prior., MPLS*
 - *Control Congest.* o *Cuello Botella o Bloq.*: *Alg.* que soluciona un colapso de Red por un exceso de *Traf.*. Ver *Colis., Monit., Gusano (Malw.), Ataq. DoS, Sobrecarg., Ethernet.*
 - *Equil. F.=Compo. Opt. Conect.*. Ej: *Display, RAM.*
 - *Prod.* (Producto): lo mejor es que el **Productor avise al Consumidor** cuando esté listo (*Interr. I/O ASincr.*). Es *Unif.* Ver *Esper., Buffer, IPC, Concurr., Econom., Biolog., Recurso, Prod. Cart.. Ataq. Gener., TAP Informat.. Model. P. Tick-Tock (Intel)*

Es *Unif.*. Ver *C. Red*, *C. Internet*, *T. Cola*, *Monit. Red*, *Buffer*, *IEEE 802.3*, *Diagr. F. Estr. Control F. Interr.*,

- **Seis Grados** de Separación (=“el mundo es un pañuelo”): tipo de *Topol. de Red* (Factor Impacto (Page Rank), Web, *Internet*, Facebook,...). Supongo considerado en EnRut. (*Dijkstra*, *IGP*). La mayoría de estas son:

- **Libre Escala** ($P(\rho = k) = k^{-\gamma}$ donde $\gamma \in [2,3]$, i.e. la prob. de que un nodo tenga k conexiones es una *Ley Pot.*)
- **Mundo Pequeño** ($L = \beta \log N$, i.e la distancia media entre dos nodos L crece log con del n° de nodos N). Ver Tab. *Rut. (Escal.)*

Ver Red *Fract.*, Tma Amistad, *Jerarq.*

1.3.5 Teoría de Complejidad y Métricas

- **Metr.** icas o *Ley Matem.: Informat.. IMPORTANTE* Revisar que todos los temas tengan al menos un ejercicio (*jframe*) de tipo a) *Num.érico*, b) *Cod.igo* o c) *Graf.o*:

- T1Inform: *Entropia*
- T9LogCirc: *Karnaugh*
- T10ReprIntDat: *TFN (Num.) y Multimedia Audio*
- T16GestProc: *Plan. Proc. y Semaforo*
- T17SOMem: *Pagin. y Segment.*
- T27POO: *Her. Polimorf.*
- T39SQL: *SQL*
- T63MedTransm: *Shannon Par Trenz.*
- T12EstrDin: *Lista Enl.*
- T13FichOrgan y T14FichOrganUso: y *VSAM, Volatil Fich.*
- T18SOGestES y T19SOGestArchDisp: *Plan. con I/O y SCAN (Plan. Disco Brazo)*
- T28TRInterrSincr: *M. Layland73 (RMS Plan. RT) y Fork*
- T38AlgRel: *Ind.*

	Metr.	ej.
Arq.	Ud. y <i>Shannon Inform.</i> , Cambio <i>Binar.</i> , <i>Q DRAM</i> , <i>MIPS</i> , <i>IPC (Pipeline)</i> , <i>Pot.</i> (Intensidad)	<i>Mem. (Compo. RAM)</i>
SO	<i>T. Resp.</i>	<i>Fragm. o Segment.</i>
L.	<i>BigO</i>	
BD		
I.	<i>Metr. Softw.</i> , M. 9 (<i>Disponib.</i>)	
Red	<i>Seis Grados</i> , <i>QoS</i> , <i>Ancho Canal</i> (<i>Medio Transm.</i>)	<i>Block Factor (Ind.)</i>

- **Truco** s: estrateg. *Creat.* (inesperada) para resolver un *Probl.*

- T. *Combin.* (*SELECT SQL*), T. *DNS*

Ver Phishing (*Malw.*)

- **BigO** (Cota Superior Asintótica): F. de como crece los *Recursos Tmp o Esp.* de un *Alg.* conforme crece el tamaño de *I. n.* Aunque Donald *Knuth* (*B., Turing,*) en 1976 *Populariza la O* y estudia a fondo *Alg.*, la *O* viene de *Ord.nung* [*Landau Hist. L.*] de *Efic.* (*Ley cuando n → ∞ Masiv.* obviando factores cte y *n* pequeñas). Permite comparar *Alg.* Ej:

- *Transf.* DFT n^2 vs FFT $n \log(n)$ que cuanto más muestras más se nota la ventaja
- *Arbol Binar.* Autobalanc.: *Busq.* en BigO($\log(n)$).
- *Ord.* Quicksort: BigO($[n \log(n), n^2]$).
- *Met. Acces. Secuenc. Busq.* *n.*

Contrasta con *TAI* (ver *Efic.*). Define Complej. *NP*. Ver *Busq.* Combin. *Estr. Dat..*

- **NP** (Non Polinomial *Probl.*): P vs NP es de Stephen Cook (*Turing* A. 1982) los más famosos son los 21 NP-Completos de Karp (*Turing* A. 1985 *Hist. L.*) que se dividen en:

- De *Graf.*: la mayoría como Clique (ver), Viajante o Travelling Salesman (no confundir con P. *Rut.* Más Corta).
- De *Log.*: como SAT
- De Enteros: Suma 0 (caso de P. Mochila *Alg. Voraz*) .
- De *Primos e IsoMorf.* de *Graf.*: claro *NP* pero no se sabe si NP-Completo (ver)

Ver *Efic.*

IMPORTANCIA: para IA por *Busq.=Consult.=Topos* IDEA: usar *Filos. MATLAB* para *Lin. Alg.* (*Cola Recurs.* con *Transf.* FFT que evita Irreduc. *Wolfram*) y probar NP vs P. Creo que los *Transf.ormers* de *Aprend. Autom.* hacen algo así en *Graf.*. IDEA: *Expr.=Parentesis=Concurr.* Esto relaciona *Algebr.* con *Alg.* y con *Anal.=Transf.* Fourier de *Alg.* IDEA: *Fulkerson NP Flujo* resuelto con fluidos (ver)

1.3.6 Teoría de Conjuntos

- T. **Conj.** : ver *Domin., Prod. Cart., Grup.*
- Op. *Conj.*: *Correspond.* con Op. *Log.* *AND = INTERSECT = ∩ OR = UNION = ∪, NOT = CONTRARIO* en *Consult. SQL*
- **Card.** : $|A|$ = nº elementos o tamaño de un conj. Ver *Rel. Tab.*
- N-**Tupla** : conj. de N elementos en el que el *Ord.* importa a dif. de un *Conj..* Ej.: **Par Ord.** o 2-Tupla o Coord. 2D $(a,b) \neq \{a,b\}$. *Prod. Cart.* . Ver *Topos Direc., Tabla, Python, Dic., Lista, Estr. Dat., Alg. Rel., Mod. Rel., Calc. Rel. T.*
- **Rel.** N-**Aria** : a) $R = \{t_i = (s_1, \dots, s_n) : t_i \in S_1 \times \dots \times S_n\}$ subconj. de *Tuplas* definidas por un *Alg.* del n-Fold *Prod. Cart.* (PC) de *n* conj. b) Variedad del espacio P. C. c) *Topos*: conj. *Tuplas* que hacen V un *Predic.=Condic.=Consult.* (Tma *Codd*) $\{t_i : P(t_i) = V\}$ ver *IA*). Ver *Binar. Tabla, Alg. Rel., R. Complet. (DML) Pullback, Calc. R. Arbol B+*
- **Correspond.** (o *Rel. BiAria.*): $f : X \rightarrow Y$ subconj. del *Prod. Cart.* $X \times Y$. Es un *Alg.* y se puede representar como un *Graf.* Bipartido. Puede ser:
 - **Unívoca** [*Unic.* Bin. *Rel.*]: \exists_1 imagen \forall elemento. del Domin.). Ej: *F.*

- No Unívoca (No *F.* o *Apl.*): da Ambigüedad o *Entropia*. Ej: Modul. *Hash.*, Conj. Cociente *Equiv.*, *Card.* N:M *BD* o *DiGraf.* Aerop. representado con M. *Binar.* Ver *Primo*.
- Iso/Mono/Epi *Morf.* : *Equiv.* Bi/In/SobrYec.. Ver *Rel. Tab., Cat.*

Ver *Ord.*, C. *CHL (Cat.)*, C. *Op. Log. Conj., Tabla, FAT, Map., VLAN*.

- *EA* (Estructura Algebraica): (A, Op, Ax) donde, A: *Conj.. Op.* sobre A int. o ext. N-Aria (*Correspond.* o *Alg.* $F(x_1, \dots, x_n) = y$ que toma N I. y sale 1 O.). Da *Clausur.* = *Compo.=* conj. min. de op. sobre A y sobre sus resultados. Ej: *Tabla de Grup.* o *Ax (Axiom.* que cumplen los op.). Ver *Dat., Tip. Dat. Abstr., Estr. Dat., Jerarq., Boole, CRC Anillo*
- Comb. *Lin.* : *Binar.* (TFN), *Cod.* con diccionario, *Medio Transm., Multiplex, FDM, Concurr. (Serie), Estr. Dat. Din., Busq. L. MATLAB (Efic.), Transf., LPC*
- MultiConj.: Generalización de C. donde se permite más de una *Instanc.* por mismo elem. (no *Unic.*). Ej: *Combin./Permut.* con Repet. Ver *DISTINCT, GROUP, SQL, Hipergraf.*,
- *Rel. Equiv.* : *Correspond.* Refl. Simetr. y Trans. Ver *Semant., Reescr.*
 - Cl. E.: conj. de todos los elem. que mantienen una R. E.
 - Repres. *Canon.* : el elemento más *Eleg.* de la Cl. E. Ver *Boole.*
 - Cociente: Conj. de Repres. Canon. Ej: espacio c., *Grup.* c., anillo c., ..
 - Ej: E. *Wolfram*, E. Analog. *Digit.* (*Muestr.*), E. Church-Turing (*Comput.*). E. *Circ. Digit.* *Boole* E. *DML*, E. *Codd.*
 - *Exten.-Reduc.*: E. *Cl.-Struct, Her.*

Es *Unif.*

- *Ord.* : *Correspond.* o *Rel.* Bin. con el mismo conj. y cuya representación es un *Graf.* bipartido totalmente conexo (dice como se rel. todos los elementos con otros).
 - O. Parc.: *Correspond.* Reflex. Trans. y Antisimetr. (si aRb y $bRa \Rightarrow a = b$).
 - **PreOrd.**: Reflex y Trans. o una *Cat.* donde solo \exists_1 morf. de un obj. a a otro b). Ver *Lattice* o *Retículo (Boole), Markov, Concurr. SPN*
 - Rel. *Equiv.*

Ver *Conoc., Alg.* O. Quicksort en *Busq.* (*Arbol B., Rot.*), *Ordanigrama (Diagr.)*, *Tupla, InDepend. Log.-Fisic.* (*SGBD*)

1.3.7 Problemas y heurísticas

- Met. *Cient.* : *Desarr. Iter., Depur.* (o *Test*) *Evol.*, feedback o retroalimentación. Se hace *Public.* revisadas por pares. No confundir Evid. (*Est.*) vs Dem. (*Test Caja Negra*). Ver Alto *Rend.*, *MATLAB, Fortran, EntscheidungsP (Turing)*
- Resol. *Probl.* : hay varias formas, todas son *Desarr. Iter.* (Met. *Cient.*): a) **Matem.**: Met. Polya, *Gener.* (ej. Cierre Poncelet mejor en 3D que en 2D con Geom. Proyect. o Tma Fund. Alg. mejor en \mathbb{C} que en \mathbb{Z}). b) **Ing.** Softw.: c) **Pensamiento Computacional**: las 3 As: *Abstr.* (formular P.), Autom. (expresar sol. en *Alg.*), Analis (ejecuc. y evaluac.) d) T24 TecnoZubia: Anal. (datos necesarios), Alg. (descripción), Codif. (en L.), *Compil.* (pasar L. Máq.) Ver *Desarr.. P. Tecn., P. Mejor Esfuerzo (QoS)*

- P. *Inv.* : Conoc. el Alg. o F. de un Sist., Blackbox o Fuente Inform. (Fisic. o Abstr.) observando sus Dat.s (T. Cat., Inform.). Esto permite Predecirlo y Tomar Decisiones. Ej: Homogr.VA, Graf. Represent. Ver Ing. Softw. I., Depur., Recipr.
- 21 P. Karp NP-Completo
- P. InDecib.
- 3 P. Compart. Recursos (Concurr.)

Ver Truco.

1.3.8 Correspondencia Curry-Horward-Lamberk

- *Equiv. Correspond.* CHL o Trinidad Computacional [ncatlab]: es Unif. Comput. (L. Lamb.)=Esp. (Topos, Cat.)=Log. (Tip.). Fig 1.7. Ver Decl. (Dem.=Progr.), Meijer

T. Cat.	T. Tip.=Log.	Lamb.)
Cat. Obj.	Tip.=Prop. o FBF (Progr.)=Dem.	L. Form. Progr.

- *Calc. Lamb.* : es un Sist. Form. para expresar Comput. basado en programar F. (Progr. Fun.) mediante:
 - *Vincul.* (Name Binding, Denominar, Definir): asociar un Id. a un Dat., Var., F o Alg.. Importa el Ambito. Equivale a HiperEnl., extender explicación (hyl). Ver Enl., Web, Def. F., Enl. Din. POO, Rel. Tab., Asign. (Alg. Rel.), Join
 - *Subst.* : lo que hacemos para resolver Sist. Ec. Ver Enl., Preproc.. Es el acto de Compo.. Ver Join, Eval., Compr. Dic., Cript.

Su importancia reside en:

- Recurs.=F. Comput. (ver Tesis Church-Turing)
- *Reescr.* o *Traduc.* . **Confluente**: saber si dos Expr. del C. L son Equiv. es InDecib.. Similar a Simplif. Algebr.. **Tma Church-Rosser** (da igual orden de aplicar Reglas de Reducción que al final se Confluye, usado T. Unif. Wolfram). Parece Semant.. Es un Unif.. Ver Interop., Interf., Transduc., Cod., Compil. vs Interpr., Autocompl., Wittg., NAT, Gateway, TLB (Pagin.)

Es para L. Secuencial a diferencia del Calc. Pi (Concurr.)

- T. *Tip.* o Log. Intuicion./Contract.: Desventaja porque no permite Reus. Cod., pero Ventaja porque permite gratuitamente Depur. o incluso checkear o Filtr. Dem. validas (ATP ver Predic.=Topos) [Milewski] (util en MetaBiolog.).
- *Cast* (no confundir con Broadcast): en L. **Fuerte** T. no se permiten violaciones de los T. de Dat., i.e, dado el valor de una variable de un T., no se puede usar como si fuera de otro T. distinto a menos que se haga un Cast. Ej: L. Pascal Ej: Python es Fuerte y no permite string+int pero C No es Fuerte y si lo permite, los Cast en C son a nivel de Compil. (quita Warnings).
- T. Dat.: los Obj. de T. Cat. lo son. Ver Biolog., MiraCast (Broadcast).

- T. Básico (char, int), T. Deriv. (*Num. Caden.*, array). Ver *C*.
- Chequeo de *Consist.* T.: ver *ATP*
- T. *Depend.*: 2 ej. a) F. D.: I. n O. vect. tamaño n. *Cpp <vector> int a[5] = f(int,5)=f* recibe tipo int y 5 y devuelve un tipo mayor [Milewski]. b) Pares D.: *Vect(n),len(vect)=((-2,0,1),3)*. Encontrar *Equiv.* entre 2 T. es *InDecib.*. En *Polimorf.* o Progr. *Gener.* el T. es parte del arg. de I.

Ver *Tip. Dat. Abstr. vs Estr. Dat., Form., typedef (Struct), Caden., Overflow*

- T. *Cat.* (the field of *Compo.sitionality* (Composicionalidad)): una Cat. *Graf.* o *Diagr.* que describe como cambios (*Morf.*) en distintos *Obj.* pueden llevarnos al mismo Obj.. Los Obj. deben poder **Compo.. Filos..** es *Conoc.* no el “en sí” si no el “en otro”.
 - *Pullback* o Prod. *Fibrado*: 2 signif. [Spivak] a) *Compo.* de Functores sobre Conj. b) Límite de Cospan ($\{(x,y)s.t.f(x) = g(y)\}$ subconj. Prod. Cart. XxY). Ver *Join, Migr.*
 - *Haz* (Teor. Haces o Sheaf): *Esp.* grande-arriba se *Proyec.* sobre uno pequeño-abajo. Arriba las curvas son suaves mientras que abajo parece que no (da saltos). Podemos Seccionar (de abajo a arriba) o Proyectar (de arriba abajo). Ej: *Fibrado Vect.* (Vector Bundle) o mas concretamente un Fibrado tangente de una variedad diferenciable que asocia un plano a cada pto. Ver *Topos*
 - *Filos.* T. C.: Sintesis (*Blackbox*) vs Anal. (partir). Ver Kay *Polimorf.*
 - Functor: ver *FQL*

Ver *Blackbox, Concurr. Encapsul., Interf., Obj. Expon., Curry, Meijer*

- *Topos* : es los “sitios o lugar” (no solo en el sentido de Lugar *Geom.* o *Esp.* donde los Inputs de una *F* o *Alg.* cumplen un Output (también un Alg. o Predic. o Intervalo Temporal) pg. 249. Ej:
 - *Rel.*: *Predic.=Condic.=Consult.=Asoc..*
 - *Predic.* \in *Tip.*
 - *Direc.* (lugar donde estás??): con una *Tupla* como *Dir.* (SO), *Id.* o *Ambito* (Alg.), *Consult.* (BD), *IP* (Red), URL (*Web*) para *Busq. Vel.* (en Sist. Referencia).

Es *Unif.* por *Direc.* (*Id...*) y *Busq.* *Binar:=Dir:=Naveg.=Opt. Consult..* Ver *Impl.antar, Haz*

$$\text{Ej1: } p(n, z) = \begin{cases} \text{true} & n < |z| \\ \text{false} & n > z \end{cases}$$

- \exists Conj. $n \in \mathbb{N}$ que cumple el *Predic.* $\forall(z : \mathbb{Z}).p(n, z)$? Sol: $n = \{0\}$
- $\exists n$ t.q. $\exists(z : \mathbb{Z}).p(n, z)$? Sol: $n = \mathbb{N}$
- $\exists z$ t.q. $\forall(n : \mathbb{N}).p(n, z)$? Sol: $z = \{0\}$
- $\exists z$ t.q. $\exists(n : \mathbb{N}).p(n, z)$? Sol: $z = \mathbb{Z}$

Ej2: Sea $E = \{n \in \mathbb{N} : n \text{ es par}\}$, $P = \{n \in \mathbb{N} : n \text{ es primo}\}$, $T = \{n \in \mathbb{N} : n \geq 10\}$ \exists Los 3 elementos más pequeños de $(E \wedge P) \vee T$? Sol: $n = \{2, 10, 11\}$

1.3.9 Lógica formal

- *Log.* :
 - L0 o de *Boole*

- L1 o de *Predic.* =*Expr.*=*Condic.*=*Rel.* (*Topos*, Tma *Codd*): Es una *F*. que da *V* o *F* según valor dado a *Var.* y no debe confundirse con *Prop.* que no tiene Arg. o *Var.* de entrada. *P.* \in *Tip.*. Tiene el mismo Poder *Expr.* que *DML*. Creada por *Frege* y Pierce *Hist. BD*. Ver DIRHA *NLP, Unic., Decl.*.
- L2 o Inducción:

Ver *Condic.*

- *Recipr.* : $T \Rightarrow H$ dado $H \Rightarrow T$. Ver *Impl.ementar* (no Implicar), *SATA, Inv., IoT* (Edge Comput.)
- *Axiom.* ver: A. *Ortog. (Cohes.) Algebr. Boole*
- *Prop.* (no confundir con *Predic.*): FBF=V (en Log1 todas tienen *Dem.* por *Goedel* 0, de hecho Tma=Prop=FBF). En Log. 2 puede no tener Dem.).
- *Tma* : *Prop.* con *Dem..* Ver T. Progr. *Estr.*, T. CAP *Almac. Distrib.*, T. *Concurr.-RT*. T. Resto (*CRC*)
- *Dem.* : secuencia *Prop./Axiom.* y Regla Infer.. Ver Tma *Shannon-Hartley*. Tercero Excl. (*Boole*), *Decl.* (*Dem.=Progr.*), *Modul.*, vs Evidencia (*Test Caja Negra, Cient.*), *ATP*, Layland73 (*RMS RT*), D. Fact. *Primos*, T. Chino *RSA*, T. *Codd*
- Tma *Goedel* : 0, 1 y 2.
 - *Consist.* (= *Coher.*): basado en Chequeo de *Tip.* (ver *ATP*). Es *Unif..* Ver *FN, Integr., Compo. Cooper.*
 - *Complet.* : Poder *Expr.* equivalente a Maq. *Turing Univ.* (*Chomsky*). Ej: C. de los *DML*. Ver *Integr. SGBD*.
 - *Decib.* : $\forall FBF \exists Dem. Autom.$, i.e. existe *Alg.* para saber si una FBF es *V* o *F*. *Probl.* Indecib.: *Tip.* Depend., Ver *Autocompl.. Enumer.*
- *Sintax.* o *FBF* o Gramática L. *Form.* o *Restr.*: permite Poder *Expr.* sin ambigüedad un *Alg..* Es un límite de *Wittg.* y Tesis *Church-Turing*
 - *Form.* (Formal, Formato o Formateo (*Sist. Arch.*) o *Tip.* o Formulario (*Apl. Web*) o **FBF** (Fórmula Bien Formada): sigue el P. *Frege*. Ej. L. *Music.* o *Expr.* Regular. Es *Unif. Equiv.* a *Estr. Dat.* y *Estr..* Ver F. Cerrada Binet *Recurs.*, F. *Instr. OpCod.* Ver *MP4, Instr. CODE2*.
 - *Pars.* ear: *Anal. Sintax.* o *Recon. Log..* Ver *Chomsky*.
 - *Nivel* o *Capa* de L.: el n. superior restringe las combinaciones (*Compo.* desde *Atom.*) posibles del nivel inferior. Ver *Recon.*
 - BNF (Backus-Naur Form): forma de especificar la Gram. de HTK (con ORs y *Compo.*). Fig. 25.8

Ver *Chomsky, Token, Atom., Form., C*

- *Semant.* (o Significado): ver *Frege* (*Semant.=F.* de *Sintax.*). También es el comport. de *Ejec.* un *Alg.* (ver *Instr.*).
 - Se encarga de conocer todos los programas que producen la misma salida para un input (el más corto es el Repres. Canónico de Cl. *Equiv.* y mide la *Entropía TAI*).
 - Conj. de Expres. F. o Códigos que **Confuyen** en *Reescr.* (*TAI*).
 - La Sem. restringe la Sint. (i.e. según el Tema se elige una Gramat. diferente de Rec. Habla). Ver *Cl.*

- ¿existe un L. *Form.* o *Traduc.* que nos diga cuando dos *Expr.* Sint. tengan la misma Semant.? Ej: 2 resistencias serie de 1 Ohm equivale semanticamente a una de 2 Ohm (aunque sus representaciones sintáticas son diferentes pg. 5).

Ver WordEmbed (*NLP*), ReFact., Model. *ER*, Ontol. Polimorf., Meta, Web S. (*HTML5*), Diagr. UML

- **Meta** Dat.: ver Dic. *BD*, PDUd., *Semant.*, M. *Biolog.* (*TAI*), Sentido Común (*Segur.*), Direc-tiva/Macro, Atrib. Fich..
 - **TAI** (Teoría *Alg.* de la *Inform.* [Chaitin] *B.*): se encarga más de la *Eleg.* que de la *Efic.* (*BigO* o *Esp./Tmp*) de los Progr.
 - *Axiom. Ortog. (Cohes.)*: “Lo bueno y breve 2 veces bueno”, “Decir lo máx. con lo min. posible”
 - **Teoria** o **Conoc.** de un *Sist.*: es el *Alg.* mínimo que lo explica. *Axiom. Ortog. (Cohes.) Cod.* Secuencia=Predecir=buscar el *Alg.* o M. *Turing* que explique de forma mínima la Sec.. Si impredecible=poca compr.
 - *Entropia*: de una *Secuenc.* o *Alg.* = tamaño del progr. min. (más *Eleg.*) en un *L.* dado.
 - *Compr.*: *Equiv.* a Entendible, Legible según *TAI*.
 - **Omega** : prob. de elegir una M. *Turing* que pare. Requiere que los programas sean *Cod.* *Pref.*
 - *MetaBiolog.* [Chaitin12]: organismos son Maq. *Turing* que Paran y que *Evol.* para ver quien produce el N° Mayor (Busy Beaver). En vez de Maq. Oráculo usar *Tip.* que gratuitamente *Filtr.* programas no válidos. Añadir *Efic.* (no solo *Eleg.*), en un mundo *Fisic.* 3D se gana mucho computando en 2D en vez de 1D (*Concurr.*).
 - **ATP** (Automated Theorem Proving) solo ocurre a pequeña escala.
 - **Asist. Dem.**: ej: *Lean*
 - **Calc. Constr.** (no confundir con C. *Rel.*): de Thierry Coquand (INRIA) como *Lean* o Coq con *Tip.* Inductivos. Comprueba automáticamente que las pruebas son *Consist.*
 - *Progr.=Dem.* (*Correspond.* CHL)¹.
 - *Test* o *Verif.* o Chequeo de *Consist. Tip.*: lo hace el *Compil.* al mirar que la *Compo.* de las *Decl.* de *F.* es *Consist.*
- Buzzard : quiere *Digit.* las *Matem.* para que la *IA* las use. Por primera vez las máquinas pueden **Razon.** (Checkear L. *Form.* cerrado) más que *Comput.* (*Proc.* Num.) gracias a la T. *Tip.* (*Correspond.* CHL), *Interop.* de los Tmas y *Modul.* de las Matem. Se puede crear un *Graf.* de *Depend.* como https://leanprover-community.github.io/f1t-regular/dep_graph_document.html
- **Creat.** (*Ing.enio*): desde [Gödel] [*Turing*] las máquinas no pueden tenerla. Ver *Truco*, Ventajas Top-Down (*Dis.*), *Proyec.*

Ver *Fake*, *Haskell*, *Progr. Fun.*, *Test caja Blanca*, *Curry*. *Autocompl.*

¹i.e los *Progr.* son teorías de una *Log. Form.*, y los cálculos son deducciones en ese espacio Log.

1.4 Otras disciplinas

1.4.1 Procesamiento de señales

- Proc. *Signal* o Señales:
 - *Dat. Perd.* : llenar/inputar *Opt.* según un modelo *Est..* El *Aprend. Autom.* se reduce a entrenar un rellenador de huecos. Ej. Sudoku puede tener varias soluc., ambigüedad (*Entropia*=Sist. Ind.), Regres.=Sist. Sobredet.), Cod. *Huffman* como???. Ej. Mem. *Asoc..*
 - *Err.* Concealment=Packet Loss (*Paq. Perd.*)

Es un *Manej.* del *Err..* Ver *Null*, *Nan*, *QoS*, *Paq.*, *Corrupt.* (*Integr.*), *Fall.*, *RAID Redund.*, *Ethernet*.

1.4.2 Física

- *Fisic.* : es obtener el *Conoc.* (*Alg.* en **Ec. Difer.**) de un *Sist.* y predecirlo desde las *Condic.* Contorno (Pos.+Vel.). La predicción es el *Opt.* a un *Probl.*. El *Sist.* puede ser *Analog.* (señales) y *Digit.* (F. *Est.*). Se encarga de la ME+ST+I (materia-Energ. en *Esp.-Tmp* con *Inform.*).
 - F. *Digit.*: IDEA el mundo puede ser una *Imag.* Vect. hace *Digit.* lo *Analog.*
 - F. Ligadura (*Restr.*, *Req.*, *Condic.* Contorno):
- Es *Unif.*. Ver *Recurso*, *Dat.*, *Conoc.. Entropia Vel. Pot., Robot, Par Trenz., C. Fisica, Impl., Hologr, Distrib./Paral., Electr, EM. Ley, Mem. Virt.*
- *Esp.* : Ver *Topos*, E. *Libre* (Asign. *Sist. Arch.*), E. Nombres (*Ambito*), E. *Direc.*, *Combin.*, *Distrib./Paral.*, *Fragm.*, *Cuota*
- *Tmp* (no confundir con *Temp.*): tiempo o temporal en *Fisic..* Ver *Fich. Tmp, Registr. Tmp. Multiplex., Fisic., Reloj, Proyec., Eficiencia, Curva Aprend., Pot., BigO, TAI, RT, Distrib./Paral., TKIP (WPA)*
- *Energ.* : “La E. no se crea ...” (Wittg.). Ver *Pot.*, *RISC*, *Mem.*, *Esper.* Activa, Secuencia. Asincr. BiEst., *SRAM*, *BLE (IoT)*, E. Prop. Comput. (*Ecolog.*)
- Min. Acción: *Refrac.*
- Espectro *EM* : ver Fig. 1.6. Se puede hacer *Multiplex.* en Frec. Permiten las *Wireless*. Ver *Medio Transm.* no guiado, *Disco HDD, Electr, Apantall.*
 - *Vel. Luz* impone límites a la propagación (*Latencia*) y *Sincr.* de las señales y la Comput. Ej: *Cond. Carr.* Ver *Fibra, Refrac., Disco DVD, Hologr, Render. (3D). Lifi*
 - *Antena* : parece un *Transcep.* puede *Transm.* (*Actuad.*) y R. (*Sens.*) modificando la Intensidad de corriente. Tiene un Ancho *Banda*. Fig. 1.6 A. 4G/5G *Movil*
 - *RF, Infrarrojo, Microondas*
 - Ultravioleta: *EPROM*

Ver *Tierra*

- *Wolfram* :

- Irreducibilidad W.: estado futuro de un sist. no predecible salvo que se corra (*Ejec.*). Malo para *Opt.* (ver). IDEA: se puede saltar con *Transf.* FFT (averiguar $y(n) = \sin(53n) + \sin(31n)$ no tengo que ejecutar hasta llegar si no computar).
- Principio de *Equiv. Comput.*: en [A New Kind of Science] la mayoría de los *Sist.* de la naturaleza que no son obviamente simples son IsoMorf. a una Maq. *Turing* Univ. i.e. Hay muchas *Arq.*, *CPU Neumann*, *Autom.* 110,... Otros: Puerta *Log.* con agua.
- *Autom.* Celular: A. 110 es una M. *Turing*. Ej. de *Emerg.*, reglas simples comportamientos complejos. Binarios y *Markov* hay 256 según *Wolfram*.
- W. L.: ver
- W. Alpha: ver *Servid. Apl.*
- T. Unif.: basada en *Hipergraf.* y Tma Church-Rosser (*Reescr.*)

1.4.3 Biología

- *Biolog.* : estudio (de la *Evol.* y funcionamiento) de los *Sist.* químicos orgánicos. Es *Clausur.* como las *EA*. Hay *subEstr.* gracias a *Interop.* *Interf.* y al uso de los mismos componentes o *Prod.* (mismos *Tip.s*): ADN-DNA (ver *Almac.* *MS*), lípidos, glucosas,... unos pueden alimentarse de otros. aunque puede innovar a otros materiales. Creo se da *Concurr.* y *Comunic.* a nivel Hardw. con Circ. *Secuenc.* *ASincr.* para consumir menos *Energ.* en *Esper.* *Activa+Interr.* (*Signal* o *Event.*)

Las *subEstr.* son *Sist. Abierto*. Es una mezcla de *Fisic.* (por trabajar con materia) e *Inform.* (por la *Comunic.* y *Proc.*). Se parece al *ATP* en la **Compl. Inf.**) y en que los Tmas son *Interop.* (los *Modul.* se hablan entre ellos):

- *Red Superp.*: Redes dentro de Redes para comunicarse.
- *Red B.*: hay muchas Ej: R. Trófica (quien se come a quien) o R. Interacción de Proteínas

Ver *BDOO*, *Coher.*, *Bioinform.*, *MetaB.*, *Turing*, *Robot NiTiNOL*

- *Evol. Biolog.*: los *Sist.* se pueden *Escal.* o *Desarr.* pero no crecen de forma repentina (*Vel. Pot...*) quizás por *Equil.* Ecológico y *Latencia*. **CoEvolución** y AutoCreatividad: dentro del un mismo organismo ej. la trompa del Mamut crece porque el cormillo lo hace pero Sin Quererlo le da una trompa larga que hereda el elefante. Ver Met. *Cient.*, E. *Abstr.* (*Hist. L.*), *Aprend. Autom.*..
- Sist. *Emerg.* : *Sist.* o *Red* donde “el todo es más que la suma de las partes”. Sus elem. se *Comunic.* con *Conmut.* *Paq.*, *Broadcast*, *Buffer*. Ver *Autom.* *Cel.*
 - Es Dis. *Cooper.* o L. *Paral.* *Concurr.* (como *NN* entrenandose T. *Ejec.*) que contrasta con Dis. Secuencial (arriba abajo izda dcha) de los L. tradicionales.
 - E.: es como la Hab. China de Searle (donde el libro contiene el Alg. de construcción) aquí las pequeñas reglas de cada Agente dan el Alg. de constr.

1.4.4 Economía

- *Econom.* : es el *SO* de los *Recursos* del planeta. Teoria de la *Prod. Efic..* Política: gestión intuitiva y sin tener *Model.* para poder *Opt.* estadísticamente.
 - Comunism.: Recursos son de todos (promueve la solidaridad). Capitalismo: recursos del 1º en adquirirlos y sacarle partido (promueve la creatividad). Perplejidad: dado que no es claro quien es quien se merece el recurso cobrar según horas trabajadas (Proudon).

- IDEA Compromiso: dado que el mayor miedo del capitalismo es la vagancia-noEficiencia y el del comunismo la exclavitud, usar las TIC InformationTechnology, para encontrar el equilibrio.
Justificación del capitalismo **Cena compartida** (somos egoistas si el daño se reparte, por eso es mejor que el daño que hagas en tu empresa lo pagues tu) vs
Just. del comunismo **Tragedia Comunes**: actuar en contra del Bien Común por la competencia.
- T. Juegos

Ver Disciplina *Cola*

1.4.5 Filosofía

- **Filos.** : los *Comput.* nacen por probl. F. de la Parada Hilbert, resuelto por *Turing*.

- A.: Genialidad *Shannon*, F. RISC MMIO.
- SO: F. Chinos (InterBloq.), *Petic..*
- L: *Kay, MATLAB, Parad. Progr., Agil.*
- BD:
- R: Parcheo *E2E* y democracia *Internet (MPLS, QoS), Pila Protoc.*

Ver *Unif. Lampert (Lista)*, *Wittg. Tip. Dat. Abstr.*, F. *Dat. Perd.*, F. *T. Cat.*, F. *Opt., Abstr.*, F. *Interr. (Prod.)*

- **Wittg.** : *Filos.*

- *Abstr.* vs *Descr.*: “Lo que se muestra, no se puede decir” “what can be shown, cannot be said” [Tractatus 4.1212] “La *Inform. (Energ.)* no se crea ni se destruye solo se *Cod. (Traduc., Impl., Progr., Simbol., y se Metr.)*” [Juan] (, *Alg., Inform.*). Falta el “se hace” =*Proc. Fisic. Ejec..*
- Poder *Expr.*: “Los límites de mi *L.* son los de mi mundo” *Form.*, M. *Turing*.

Ver *Hardw., Instanc., Eval., Proyec.*

1.5 Impacto de las (nuevas) tecnologías

- **Ecolog.** : forma parte de la *Instal. Segur.* Se rige por la *Ley (Estand.)* de tratamiento de RAEE (Residuos de Aparatos Electrónicos y Electrónicos). Ej:

- *Energ.* proporcional a la *Comput.*: para que *Centro Dat., Comput. Nube o Alto Rend.* más *Efic..*

Ver *Europ., Display LED*

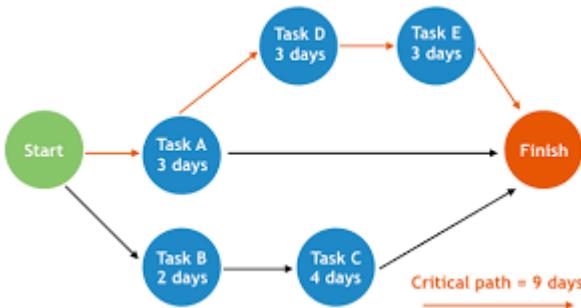
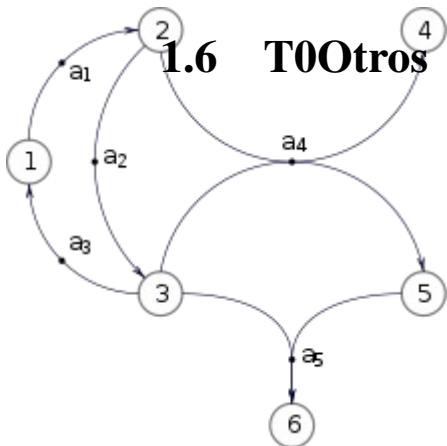
- **Tecn.** : Impacto y *Probl.*

- Sedentarismo
- *Fake*
- Ciberacoso, adicción, *FakeNews*, falsas-esperanzas

Converg. *Tecn.:*

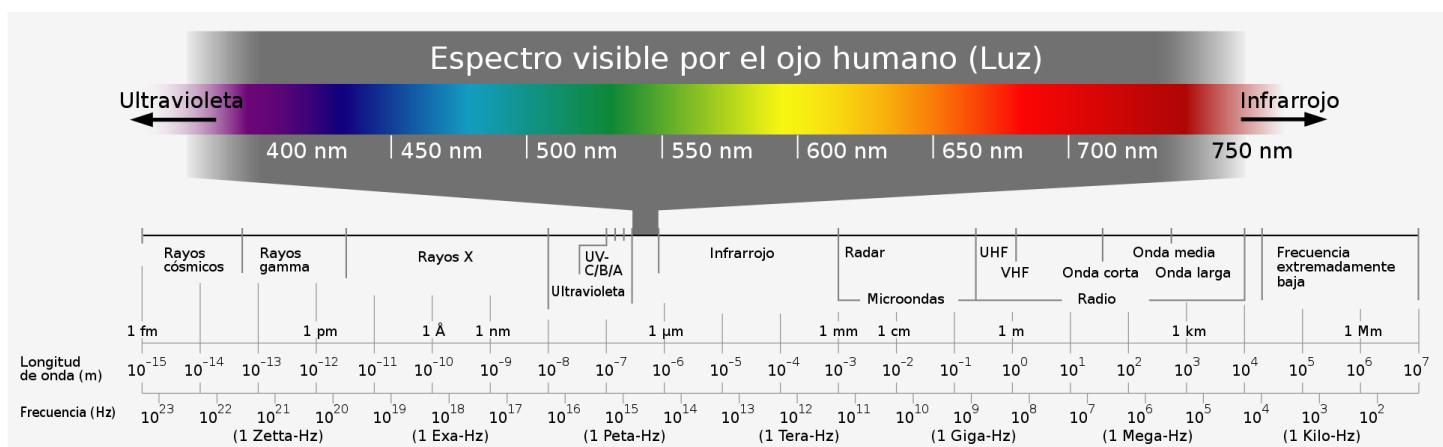
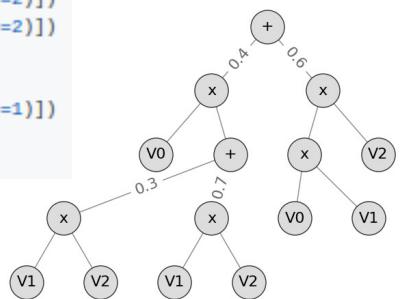
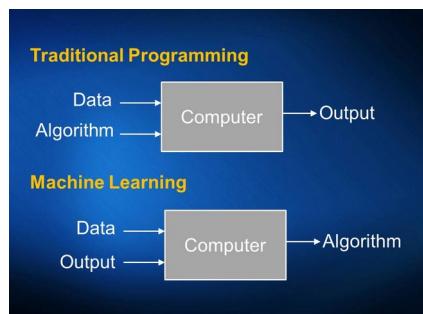
- *Music.=VideoClip=Realidad Virt.*
- *Triple Play. y SmartTV*
- *Home Rut.*
- *Vehiculos=máquinas=Robots*
- *Realidad Virt.=Realidad Aumentada*

Ver Converg. T. *MS (Flash, Opt...)*, *Legacy*, T. *Integr. Moore, Resol., Impr., Empresa, Protoc.* vs Estand.

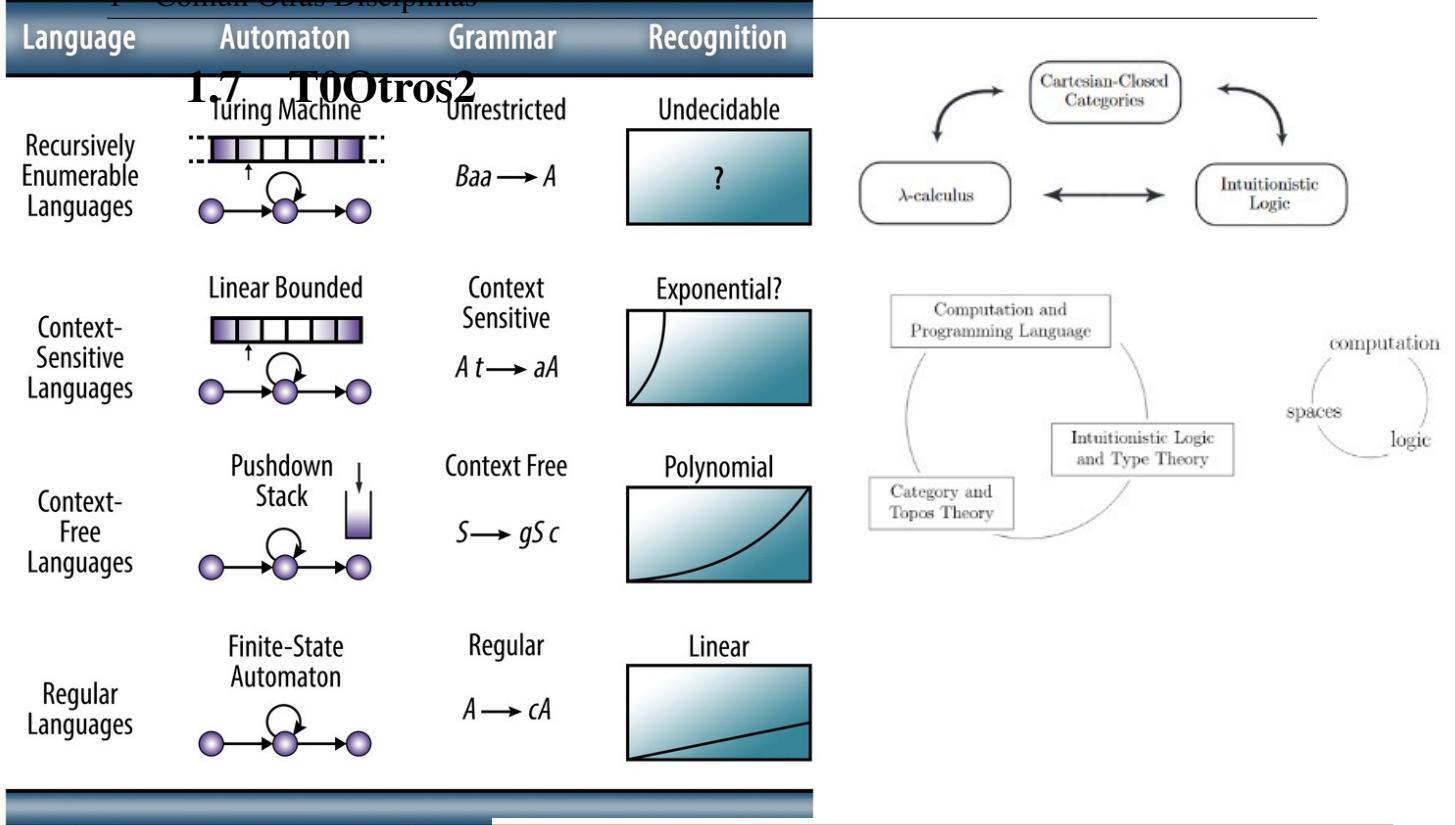


```

p0 = Product(children=[Categorical(p=[0.3, 0.7], scope=1), Categorical(p=[0.4, 0.6], scope=2)])
p1 = Product(children=[Categorical(p=[0.5, 0.5], scope=1), Categorical(p=[0.6, 0.4], scope=2)])
s1 = Sum(weights=[0.3, 0.7], children=[p0, p1])
p2 = Product(children=[Categorical(p=[0.2, 0.8], scope=0), s1])
p3 = Product(children=[Categorical(p=[0.2, 0.8], scope=0), Categorical(p=[0.3, 0.7], scope=1)])
p4 = Product(children=[p3, Categorical(p=[0.4, 0.6], scope=2)])
spn = Sum(weights=[0.4, 0.6], children=[p2, p4])
    
```



1 – Común Otras Disciplinas



Automata theory

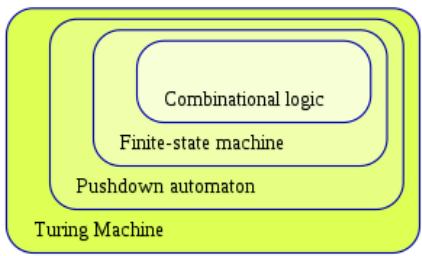


Table 1. Examples of logical and statistical AI.

Field	Logical approach	Statistical approach
Knowledge representation	First-order logic	Graphical models
Automated reasoning	Satisfiability testing	Markov chain Monte Carlo
Machine learning	Inductive logic programming	Neural networks
Planning	Classical planning	Markov decision processes
Natural language processing	Definite clause grammars	Probabilistic context-free grammars

Chapter 2

Común Arquitectura

- **Electr.** : Ver *Circ., EM, Imped., Pot., Transistor, Dispos., Tierra, Apantall., BD, Fisic., Shannon, IEC ISO, Cable, Signal, PLC Red, Rack, RF. SAI (Disponib.), DNIE (Certif.), Perif.*
- **Circ.** Camino *Graf.* cerrado.
 - C. Electr.: es un *Hipergraf.*..
 - C. Combin. o Secuenc.
 - C. Red: es un *Recurso*
 - *Conmut.* C.:
 - C. Virt. (*TCP*)

Ver *Arbol, A. Recubr. (STP), Recurso, GoTo, Equiv. C. Digit. Boole*

- **Ley Moore** : por Gordon E. Moore (*Intel*) cada 2 años el doble de *Transistor Integrados*. Crec. Expon.. Ej: ver Tab. *RAM*. L. Huang (cada 2 años la performance *GPU* más que se dobla cada 2 años). En general ocurre en cualquier *Vel.* de *TAP* por Ley de los Rend. Acelerados Tecn.

Actual. los *Micro Intel* rondan los *5nm* y *9Giga* aunque en 2022 *IBM* ha alcanzado los *2nm* (10 veces el tamaño de 1 atom. Silicio=*0.2nm*) <https://insidetelecom.com/what-reaching-the-size-limit-of-the-transistor-means-for-the-future/> Ver Tr. *Popular Masiv., Superescalar.*

- **Transistor** : Desarrollado por Bardeen, Schottky y Brattain después de intentarlo y abandonar la *Empresa Bell Lab Hist. A..*
 - **MOSFET** (Metal Oxide Semiconductor Field Effect Transistor): Transistor MOS de FET. Los de *RF* desarrollaron las *Wireless*. Hoy *3D* para aumentar I [Noel] (*HBM*). Ver *DRAM, NAND, Flash*
 - **Integr.** (no confundir con I. *BD* o *Segur.*): dist. en nanometros entre componentes o canal. *Actual.* <14nm cuesta (ver Tick-Tock *Intel*) A más pequeño menos consumo *Energ.* y menos *Temp..* Ej: *Tecn. VLSI* (Very Large Scale I. I. a muy gran escala). Sol. [Noel] Transist. *3D (HBM)* más intensidad menos *Temp..* Ver *Microproc, FPU, iGPU, iMC*. Ver L. *Moore*
 - **Integr. 3D:** ver *HBM*

Ver *Flash*

- **Imped.** (Impedancia): n° C mezcla de $R, L, C = \text{Filtr.}$. Los *Medio Transm.* y *Sens.* (como el de gasolina) 100Ω (Ohm) como mucho. Ver *Aten. Pot.*
 - Adapt. I.: hacer que la I. de la *Signal* salida (fuente de alimentación o amplific.) sea igual a la de la carga (motor o altavoz)
- **Serie** : en *Secuenc.*
 - *Registr.* SIPO
 - Marshalling (*JSON, Persist.*)
 - La Comunic. S. lleva *CRC*.

Ver *Serie 232, PPP, Bus, SATA, Lin., Persist., USB (Arch. Dispos. ttyACM0), Lin. Concurr., S./Paral. Conect..*

- **Arduino** (Microcontrolador): ej. de sin *SO* ni *Multitar.. I/O Modul.* PWM. *Bloq.-Visual* o C. Ver *Ensambl.(RT)*.

Chapter 3

+*1. Representación y comunicación de la información. (27DicMarcos)

3.1 Introducción. Conclusión

- Ej: motivador *Presentación*: ¿porqué *Digit.* para *TAP?* ¿*Tip.* datos, *IEEE 754?* ¿Qué son *Formatos Unicode, MP3?* ¿Qué es 001? los *Estand.* de fabric. lo definen con sus *Syntax*.
- Ej motivador comunicación: ¿porqué se *Modul.* y los *Err.?*
- Ej: ¿Como almacena en *Mem.* Interna vs Externa o transmite los *Dat.?*
- *Hist. A.:* Leibniz (*Binar.*), Leonardo Torres14 y Zuse (*IEEE 754*), Nyquist (*Digit.*), Shannon, IBM (tarj. perforadas), Thompson (*Unicode*), IEEE 754, Domingos (ordenadores son extractores de *Conoc.*)

3.2 Datos, información y conocimiento

- *Jerarq. Dat.-Inform.-Conoc.:* supongamos que queremos teorizar un *Sist.* (*Fuente Inform.* o *Experim. Aleat.*) y observamos sus *Atrib.* con *Sens..* Esto es una *Fuente Inform.* que emite *Simbol.*
- *Dat.* um: elemento *Atom.* de entre un conj. *Simbol.* (*Analog.* o *Digit.*) obtenido al observar un *Sist.* (o un *Atrib.* de este). Si *Digit.* es *Obj.* manipulable (*TAP*) por un *Comput..* Si se interpreta Wittg. aporta *Inform.*
 - *Estr. Dat.* vs *Tip. Dat. Abstr.*
 - Data (vs Datum): conj. de D.

Ver PDUD. (C. Apl.), Var., D. Perd., MetaD., Centro Dat.

- *Inform.* : Conj. *Dat.* organizado (*Estr.* según un *L.*) que cambian el estado de *Conoc.* sobre un *Sist.* de quien lo recibe (*Actual.* respecto al pasado *Recurs.*). Puede ser *Analog.* (surco del disco) o *Digit.* (*Binar.*) pero es un mostrado Wittg. (“la I. solo se *Cod.*”).
 - Proc. I.=Aprend. Autom.

- Metr. o Ud. I.: *KB.. ver Binar.*

Ver *Alg., Estand., Shannon, Cuant.*

- **Conoc.** : *Alg.* (=Ec. Dif. *Fisic.*) o resumen Organiz. (*Estr.* o *Teor.* o *Ley* o *Model.*, *TAI*) obtenido de la *Inform.* de observar *Dat.os* de un *Sist..* Permite Predecir o Tomar Decisiones
 - IMPORTANTE: *Aprend. Autom.=Proc. Inform.*: (Domingos) $O = \text{Alg.}$ o *C.* sobre *D. Masiv..*
 - *Estr.* consiste en hacer un *PreOrd.* o *Diagr.* Hasse o *Map.* Conceptos (*Cluster* o *Clique Graf.* (con cortes log-max)). Fundamentos/Bases/*Axiom.*/Teoria están abajo. Ej: $\text{Poset} \subset \text{Ord.}$ (más abajo).
 - IDEA: añadir una extensión a Latex o a *Web* permita obtener un *PreOrd.* del *C.* (que permita ambig.)

Ver *Probl. Inv. Fisic., Wolfram.*

3.3 Digitalización

- *Digit.:* *Muestr. Cuantiz.*, 3 ventajas *Shannon*

3.4 Teoría de la Información

- Claude *Shannon* (*Hist. A. Hist. SO*): sus 2 geniales ideas fueron a) *Boole* o *Digit.* (ver) y b) *Entropia* (*T. Inform.* 1948 en *Bell*). Además de c) Tma S. Hartley *Canal*. Ver *Electr.*
- **Fuente Inform.** : *Model. Est.* puesto en forma *Gener.* que emite *Dat.s* a una *Vel.* Bits o *Simbol.* PS. La Correl. de las Var. *Aleat.* puede ser *Tmp* (sonido), *Esp.* (imagen) o *Alg.* Es un *Sist. a Conoc..*
 - Espac. Muestral $\Omega = \text{Conj.}$ de todos los Suc. Elem. de un Exp. *Aleat.* **Evento** o Suc. Compuesto E_i : cualquier elemento resultante de aplicar Op. *Conj.*

Ver *Dat., Blackbox*

- **Entropia** Inform. (Tma Codif. Fuentes *Shannon*): cantidad de incertidumbre o de *Inform.* ambigua o de *Redund.* (por *Correspond.* no Univ. de una var. aleat.) **Media** de una *Fuente Inform.* F . $H(F) = \sum p_i \log_2(1/p_i) = \sum p_i N_i = E[\text{Info}] = \bar{N}$ donde $p_i = f_i/n$ es la *Prob.* que salga una *Palabra i.* Equivale al n° promedio de bits del código mínimo (tamaño palabra según probabilidad [Shannon]): Ej: Codig. *Huffman*, Morse.... (con apl. a *Err.??*) La Fig. 3.11 muestra la entropía de un **Ensayo Bernoulli** (es máx. con $p = 0.5$, mayor incertidumbre sobre el resultado (Inform. Fisher Binomial). Ver *TAI. Dat. Perd.*,
 - *Cod. Huffman* : son *Cod. Pref. Opt.* y la *Entropia* mínima sale de un *Arbol Binario* de palabras ordenadas por frecuencia. Mejora al de *Shannon-Fano* que a veces no da Cod. Opt. Ver *Compr.*
 - *E. Fisic.:* se basa en una F. Aritm. ($f(n)$ =formas de hacer una tarea). Ambigüedad en *Correspond.*
- *Canal* (Tma *Shannon-Hartley. A. Banda*

3.5 Codificación

- *Símbol., Cod., Cod. Pref.*

3.6 Representación de números, texto y multimedia

- *Num.:* TFN, Enteros, Reales (mantisa), *BCD*
- *Texto:* *ASCII*
- *Multimedia:* Audio...

3.7 Comunicación y tipos

- *Transm. (Latencia, Medio Transm. caract. Duplex, No/Guiados..)*
- *Comunic.* (no confundir con *Transm.*): intercambio de *Inform.* (y *Recursos*) en ambas direcc.. Un *Sist. C.* tiene un T. (Transmisor, Sender, Emisor o Transmisor) y un R. (Receptor o receiver) a través de un *Medio o Red*, siguiendo un *Protoc.* o *Sintax..* El R. hace las Op. Inv. Fig. 3.11.
 - *Medio C.:* sist. *Físic.* por donde viaja la *Signal* o *Inform.* Ej: aire, papel, *Cable*... Vulgarmente M. *Comunic.* son el periódico, radio, TV e *Internet*. Es un *Ambito de Influencia de Capa*. Ver *MAC IPC (Mem.)*. Mem. *Compart.-Concurr.*
 - Pasos de C. *OrCo* (ver)

Es un *Unif..* Ver Mem. *Compart., Interf., Driver, IPC. C. Sincr., Canal C.*

- C. *Serie* (lleva *CRC/Paral.* (antigua): ver *Conect.*
- C. A/*Sincr.* : tipos según T.
 - S.: la *Signal* de S. puede ir en linea independ. (*Reloj*) o con *Cod.* autosincron. (el dato lleva *Signal* de *Tmp* como *Manchester* de antiguo *Ethernet*). Ventaja.: más rápida por no tener bits de arranque o sincronia (OJO en Circ. *Secuenc.* distinto). Ej: en *Paral.*
 - AS.: es un tipo de *Cod. Pref.* donde E. y R. se sincron. cuando se envian un símbolo. Ventaja: más simple por no tener *Reloj* Ej: en *Event.* teclados, ratón Ej: *I/O AS (Interr.+ No Esper. Activa, Biolog.)*. Ver *Cod. AutoS, AJAX*
NOTA: no confundir *Cod. Sincr. AutoReloj* ([*Self-clocking Signal*]) como *Manchester* vs *Cod. Pref.* que incluye los C. *AutoSincr.*

Ver S. *Perif.*, Control *Flujo Datos, IPC, Concurr., Luz, Cond. Carr., Push Git, Secuenc., SDRAM, AJAX, S. Condic. (Excl.)*

- C. Simplex/Semi/Full*Duplex* : en un *Sist. de C.* (con Emis./Recep. como walkie talkie) indica el sentido (*Direc.*) 1/2 (no simultanea, *Transcep.*)/ 2 (simultanea ej. WDM *Fibra*). Para *E2E*. No confundir con *Multiplex..* Ver *PPP, FDDI (Fibra), Multiplex. Onda, Medio Transm.*

3.8 Elementos de un sistema de comunicación

Elementos de Comunic. Fig. 3.11.

- *Digit.* y *Cod.*: ya hemos hablado.
- *EnCRIPT.*: *F. Unidir.*, Cript. *Simetr.*, *TLS*, *Certif.*
- *Compr.*: Sin pérdidas: *ZIP*. Con pérdidas: *JPG (PCA)* y *MP3*

3.9 Problemas en la transmisión

- *Distors.*, *Diafon.*, *Interfer.*, *Apantall.*

3.10 Manejo de errores en comunicación

- **Checksum**: pequeño *Bloq.* Dat. extras (*Redund.*) añadidos a un bloque mayor (al *DUD.*) para comprobar *Integr.* del mayor (si estos han cogido *Err.* en la *Transm.*). Esto **Incrusta** (Inyección o Inclus. Canónica) los datos en hiperdimensión mayor y los separa más espacialmente, si la **Prob.** *Err* es mantenida los datos se quedan dentro de hiperesferas que no se tocan haciendo que no se **confundan** y se pueden *Recon.*. Ej: Paridad, *CRC*, *Hash*. Ej: *Comando* para ver si *Imagen Disco ISO OK* descargada. Ver *AcRe*
- *Manej.* de *Err.* (*Fall.*): Se debe al *Ruido* o al *Canal* y puede hacer inutil (no *Transpar.*) la Transm. Se puede medir en *Prob.* y distinguimos Det. y Corr. Se basa en separar espacialmente más los datos (ver *Checksum*). Van en *C. Enl.* (*LLC*) y *C. Transp.*.
 - Bit Paridad (par o impar): solo det. si $\leq 1b$ de cambio. Mejora con paridad vertical 2D: agrupar Bytes en filas (con paridad horizontal) y añadir fila de paridad vertical (esto corrige???)
 - Met. **Hamming**: det. y corr. con más *Efic.* que paridad. Se basa en poner bit de paridad en posiciones potencias de 2 (1, 2, 4, 8, ...) y mezclarlos con los datos $p_1, p_2, d_1, p_3, d_2, d_3, d_4, p_4, d_5, d_6, d_7$. Usado en *Mem..*

Ej <https://www.geeksforgeeks.org/hamming-code-in-computer-network/>
 Sea 1011001 → 101R₈100R₄1R₂R₁
 R_1 hace paridad con todas posic que terminan en 1: 1, 3, 5, 7, 9, 11 = 4 + $R_1 \rightarrow R_1 = 0$
 R_2 con las que tengan un 1 en 2^a posición: 2, 3, 6, 7, 10, 11
 R_4 con las que tengan 1 en 4^a pos: 4, 5, 6, 7
 R_8 con 5^a pos: 8, 9, 10, 11:

0110101 pasa a 10001100101, si esta muta a 10001100100 se puede corregir

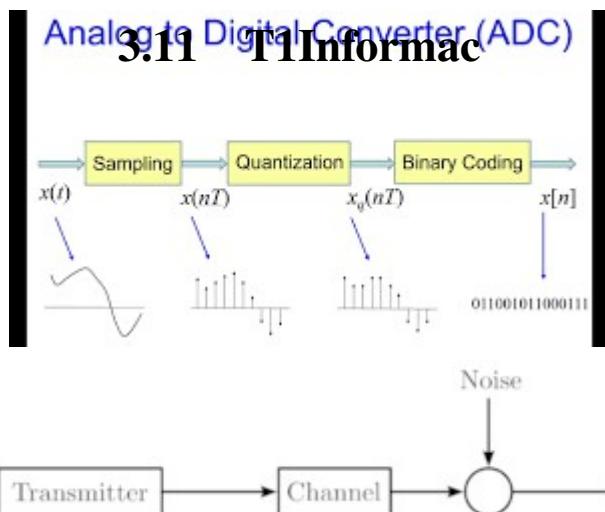
- Cod. **CRC** (Cyclic Redund. Check, Verificación por Redundancia Ciclica): Basado *Tma Resto*, en añadir un *Checksum* residuo (o módulo de dividir polinomios, *EA* Anillo Pol. sobre un \mathbb{K} : $\mathbb{Q}[x]$). Div. Hardw. Vel. con XOR (*Op. Log.*). Para detectar errores en **ráfagas** (consecutivos), **va en toda Comunic.** *Serie* como *GSM Movil*, *USB*, *Ethernet*, *Bluetooth*, *GZIP*, *SATA*, Caract. *RAM*

Cualquier *Inform.* k bits es un pol.: $10010110 \rightarrow M(x) = x^7 + x^4 + x^2 + x$

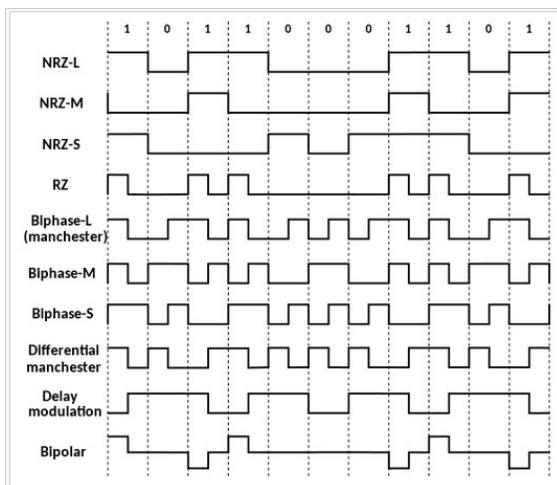
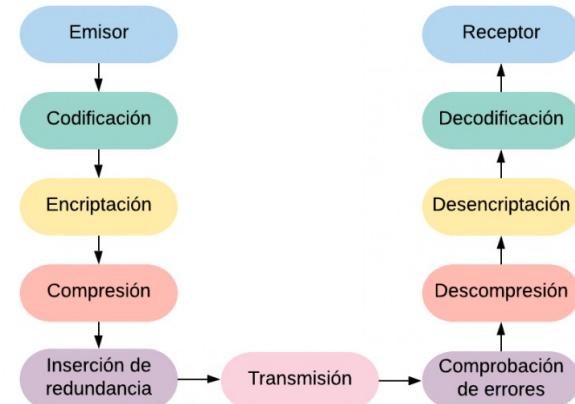
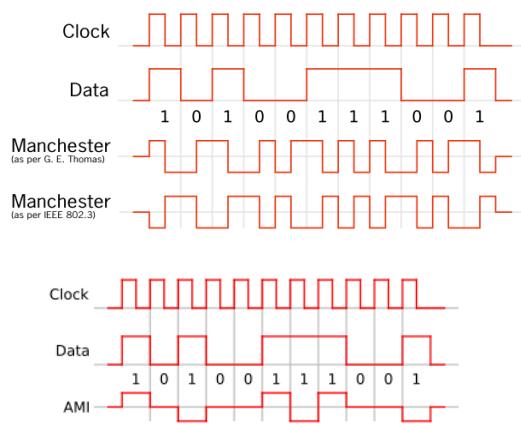
- Añadimos r bits: $x^r M(x)$
- Divid. por $G(x) = x^{16} + x^{15} + x^5 + x$ Estand. CRC-CITT: $\frac{x^r M(x)}{G(x)} = C(x) + \frac{R(x)}{G(x)}$
- Restamos $R(x)$: $T(x) = x^r M(x) - R(x)$ y *Transm.* $T(x)$
- Si en receptor $T(x)$ no div. por $G(x)$ hay *Err.*

- Cod. *Huffman*??
- Otros: Dat. *Perd.*, *Aprend.* *Autom.* (*Recon.*) y Sol. a *Ruido* (ver *Filtr.* Sal y pimient para Impul....)

Es *Unif.* Ver *Fall.*, No intencionado (*Segur.*, *Ataq.*, *Permiso*), *SSD*. Exploit (*Malw.*), *Integr.*, F. *Pagin.*, *Calidad*, *Manej.* Excep. F. *Segment.* (*Vulner.* *Overflow*)



DECIMAL	BCD	Excess 3	Aiken
8	1000	0011	2421
4	0100	0111	0100
2	0010	0101	0010
1	0001	0100	0001
0	0000	0011	0000
9	1001	1100	1111



Chapter 4

+2. Elementos funcionales de un ordenador digital. *Arquitectura (22SepJavier)

4.1 Introducción. Conclusión

- Ord. *Cuant.*, *Servid.* Fig. T2 y T3

4.2 Ordenador digital vs analógico

- **Comput.** (*Proc.* u *Ord.*): *Sist.* que corre cualquier *Alg.* (*Equiv.* a M. *Turing Univ.*) (como Computar, Ordenar o Procesar). Hay muchas *Arq.* según *Equiv. Wolfram* (señal eléctr., bolas, ver *Torres y Zuse IEEE 754,...*) pero entedemos que son *Digit.* (no *Anal.*). Se diferencia del *Razon.* y de la *Creat..*
 - Elementos Funcionales: *Bus* en centro, *CPU*, *Mem.* y *Puertos I/O*. Ver Fig. 4.10, Sinónimo de *TAP*. Ver C. Frontera (Edge, *IoT*), C. *Nube*.
- C. **Analog.** : La C. **Real** BSS es una idealización de la C. A. (ver *Fisic.* *Analog.=Digit.* con *Imag.* *Vect..* Ej. *Hist. A..* [Veritasium])
 - *Circ. A.:* amplif. operacional para integrar.
 - C. Real: Maq. BSS (Blum) que usa *Registr.* con n° \mathbb{R} (con Inf. precisión).
 - Bush Analizador Diferencial (ver *Digit.*)
 - Leonardo *Torres* y Quevedo 1852-1936: *Comput.-Calculadora Analog.* basada en "fusee sans fin" (*IEEE 754*) y "El Ajedrecista" 1912 que a diferencia del Turco tomaba decisiones *Autom.* considerándose el 1er *Videojuego* de la historia
 - Torres 1914 propone una FPU para su *Calc.*, Konrad Zuse 1934 de Berlin la incluye en la 1ª *Comput.* programable, el estandar sale en 1985 pero en 2008 y 2019 se ha revisado.
 - Zuse creó el Z3 en 1941, 1er *Comput. Boole+FPU Progr.* aunque parece no era de Propos. General (*Turing*).

Tiene más poder *Expr.* que *Turing*. Contrastá con C. *Digit..* El Conj. Mandelbrot es no computable en BSS y no se sabe si es C. en Anal. Comput. (su complemento es C. *Enumera.*). Ver *Boole Shannon*.

- *Signal A.:* Ej: salida *Sens. Fisic.* Ej: *Modulac.* FM/AM del *Telef.* o *TV* que viaja por el *Medio Transm.*

Ver *Opt.*

- C. *Digit.* ventajas de Shannon:

4.3 Historia de generaciones

- *Hist. A. :* 5 Gener.
 - 1^a 40-50: válvulas (gran volumen, no *SO* y L. Maq. *Ensambl.*). Ej: ENIAC45 (1º *Propos. general Digit.*)
 - 2^a 50-60: *Transistor*
 - 3^a 60-70: *Chip* (*Circ.* integrado (antes cables unían componentes), *Multitar.*, *Spool-ing*)
 - 4^a 70-80: *VLSI* (*Integr.*, CPU en 1 solo circ.)
 - 5^a 80-00: PCs (de *IBM*, 4GL *Hist. L.*)
 - 6^a 00-20: *Internet, Movil*
 - 7^a 20-..: *IA*

4.4 CPU

- Arq. *CPU: Neumann vs Harvard*
- Elem: *Registr., ALU, Ud. Control, Bus Sist.*
- Caracter. *CPU: Palabra, MIPS/MFLOPS/Freq. Reloj y MP, MS y Cache*
- Funcionamiento *CPU (CODE2)*

4.5 Memoria

- *Mem., Direc., Latencia, Jerarq. Mem.*

4.6 Periféricos

- *Dispos., Perif., Interf. o Conect.. Driver.*

4.7 Bus, placa madre

- Placa Madre, Bus del Sistema
- Otros: MMU, iGPU (ver), VRM (Pot.)

4.8 Paralelismo a nivel de nº de procesadores: Taxonomía Flynn y memoria compartida

- Tax. **Flynn** : Paral. a nivel de *Multiproc.* o *MultiCore*. Fig. 5.14 [Tanenbaum http://www.edwardbosworth.com/My5155_Slides/Chapter13/Multiprocessors_01.htm].
 - SISD (Single Instr. Single Data): Ej: *Neumann* y *Pipeline* o *Superescalar 1-Core*
 - SIMD (Single I. Multiple D.): la CPU procesa las Instr. 1 a 1 donde los Operandos son *Array*. Ej: Proc. *Vect.* como los de *Supercomput.* Cray o *Matr.*, *GPU* o *Superescalar 1-Core* pero op. vect. Ver *AMX Exten.*
 - MISD (M. I. S. D.): casi no hay, quizás para procesar el mismo dato con varias I. distintas o iguales y comparar resultados (por *Robust.* en vuelos de aviones)
 - MIMD (M. I. M. D): hay 2: a) *Multiproc.*: con varias *CPU* que *Compart.* Mem. Ej: *Superescalar MultiCore*), NUMA o UMA b) **Multicomput.**: varios *Micro Distrib.* cada uno con su Mem. y comunicándose por *Red*. Ej: COW *Cluster Of WorkStations* o *Grid*,).

Ver Alto Rend.

4.8.1 Memoria compartida: problema de la coherencia

- Probl. **Coher.** Mem. (=Consist.): en una *Jerarq. Mem.* (con Mem. *Compart.*) es *Actual.* sin conflicto (*Interfer.*). En Nivel lento las copias de datos modif. en Nivel rápido. Ej: C. Cache ej. SMP o UMA (*Multiproc.* o *DMA*): L1 (*Actual.* la MP con su contenido (por *MMU??*). Los Model. son MSI, MESI,... Es *Unif.*. Ver *Concurr.*, *Transac.*, *Git*, *Biolog.*, *Cooper*.
- Mem. **Compart.** : en *Multiproc.* Hardw.:
 - **UMA** (Uniform Memory Access): ⊂ SMP. Es un MIMD *Multiproc. Flynn*. Ver GDDR SDRAM
 - **NUMA** (Non Uniform): cada Proc. tiene acceso y control *Excl.* a una parte de Mem.. Es un MIMD *Multiproc. Flynn*. Ver *Bus HyperTransport*
 - **SMP** (Simetr. *Multiproc.* or Share Memory *Multiproc.*): tipo de **UMA** donde todos los Proc. Mem. *Compart.* debiendo cuidar la *Coher. Cache*. Usa *Esper.* Activa. Ver Hyper-Threading (*Superescalar*). *MultiHilo*, Hardw. Peterson *Excl. Mut.*
 - A nivel Softw. tenemos: *IPC-Concurr.* en *Unix+ POSIX* (*shmat*, *shmctl*, *shmdt*, *shmget*)
 - Otras: como Solución Paso *Mens. Polimorf.*, Alg. Peterson (*Excl. Mut.*)

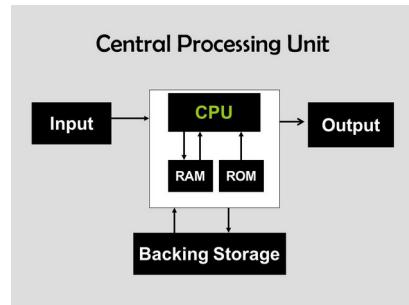
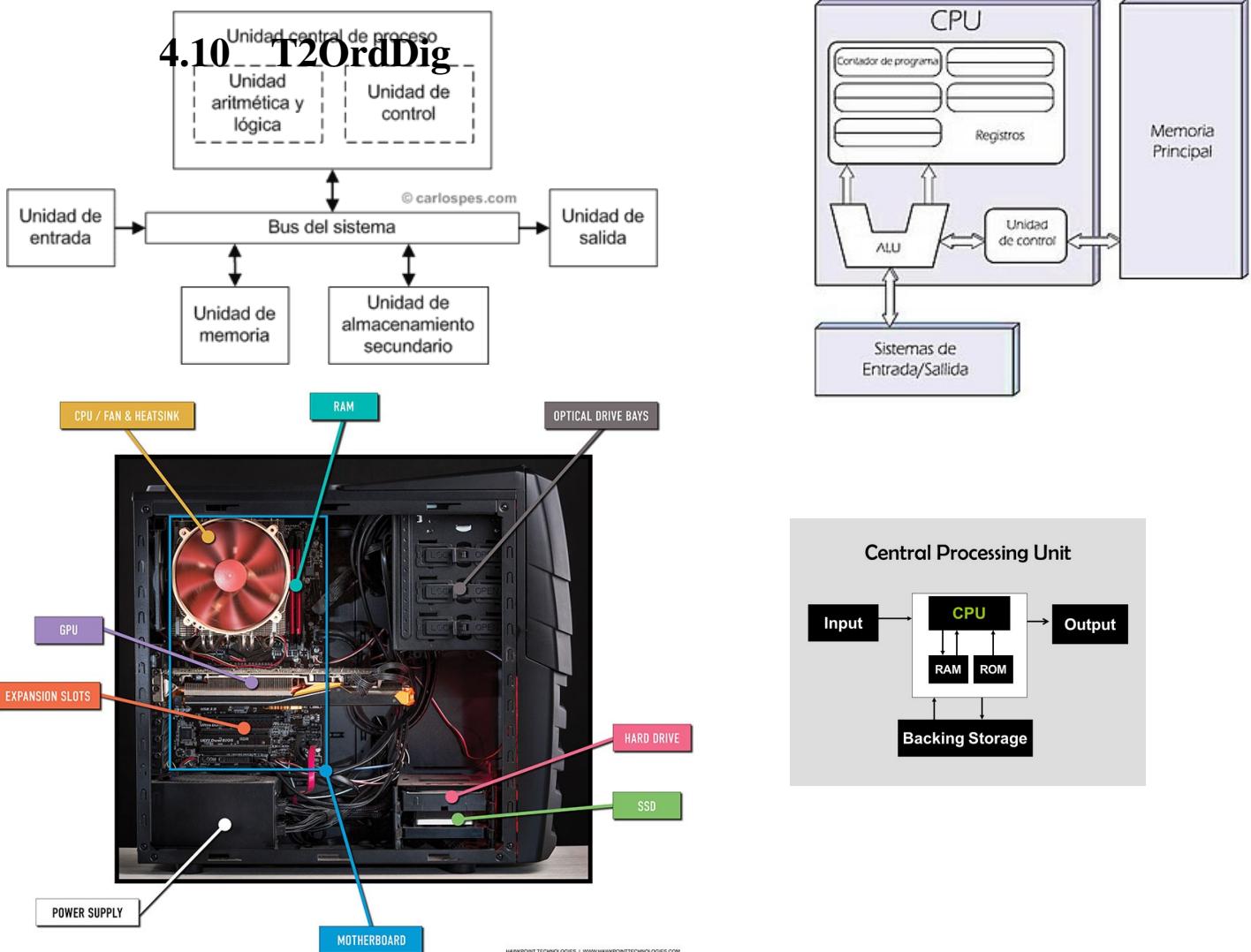
IDEA: Los 3 *Probl.* por C. *Recursos TAP* son:

- P. *Concurr.* en *Proc.*: Sec. Crit.. T. Compart. *CPU* (Esp./T. *Fisic.*).
- P. *Coher.* en *Almac.*: *Transac. MP C.*
- P. *Colis.* y *Congest.* en *Transm.*: *Canal o Medio (Red Bus)*, los 4 de *Servic. Dir.* (*Impr.*, *Fax*, *Fich....*).
- Otros: *Econom.. Trag. Comunes.*

Es *Unif.* por *Concurr.*. Ver T. C. (*Multitar.*), *Netware*, *Hub*, *Colis.*, *Red Bus*, *Permisos* y *Grup.* de C., *Arch. C. (Ln)*

4.9 Tipos según rendimiento

- *Comput.* según:
 - *Rend.: Movil.. Supercomput.*
 - *Propos.: General o Específico*
 - *Arq.: Serie, Paral.* interno o Paral. externo



Chapter 5

+3. Componentes, estructura y funcionamiento de la Unidad Central de Proceso. (5NovMigue)

5.1 Introducción. Conclusión

- ¿Mantener Ley *Moore*? Sol. *Superescalar*
- Hist. *Neumann, Torres* FPU
- Acuerdo del 82 (*x86*).

5.2 Arquitecturas

- *Arq. Comput.*: *Descri.* de un *Sist. C.* (*Estr.* u organiz. sus elementos y *Op.* que realizan) pues hay muchos según *Equiv. Wolfram*.
- *Arq. Von Neumann* 45 (*Hist. A.*): Comput. de **Progr. Almac.** en *MP* en el que hay *Cuello Botella*. Mantenida casi igual hoy. Ver Fig. 5.14.
 - 4 elementos:
 - a) Ud. *Proc.*: con *ALU* y *Registr.* de Proc.
 - b) Ud. *Control*: (*CU*) con Registr. de *Instr.* y Contador Progr.
 - c) *MP*
 - d) *Dispos. I/O: MS,...*
 - *Cuello Botella* o *Congest.* o *Bloq.*: una *Op.* de obtención de *Instr.* y una operación de R/W *Dat.* no pueden ocurrir al mismo tiempo ya que comparten un *Bus* común. Esto limita *Vel.* y *Traf..* A. *Harvard* y *Cache* lo solventan. A pesar de todo, *Actual.* es la A. dominante. Ver *Latencia CAS. Buffer, DMA*. Ver *Ind. Factor Block*
- *Arq. Harvard* : Comput. de **Progr. Almac.** (como *Neumann*) pero sin *Cuello Botella*, gracias a separación *Dat.* e *Instr.* tanto a nivel de Señales como a nivel de *Almac.* de ambos (Fig. 5.14). Esto potencia el *Paral.* y la Multiprogr. (*Multitar.*). Ej: *DSP* y *Exten. C* para *Embeb.*

- **A. H. Modif.**: *Actual*. los que dicen ser H. son H. M. Esto se ve reflejada en el diseño de *Cache L1*, se parte en L1 datos y L1 instruc (Fig. Caches). Es Neumann donde los Dat. e Instr. tienen distintas Cachés (los contenidos de la memoria de instrucciones son accedidos como si fuesen datos). Ej: *x86*

5.3 CPU. Características

- **CPU** (Central Proc. Unit.) (no confundir con *Micro* o *GPU*): el cerebro, encargado de *Ejec.* un *Progr.* (ejec. y control de *Instr.*). Tiene Max. Poder *Expr.* gracias a *ALU Recurs.* (*Chomsky*). Ver *Multiproc.*, *L. Moore*
- Elementos CPU: *Registr.* Fich. vs R. Internos, *Bus Sist.*, *ALU*, Ud. *Control*, *Jerarq. Mem.*
- Caract. Básicas (las del *Micro*): L. *Palabra*, *Reloj*, *MIPS*, R/CISC. Otras MultiCore..

5.4 Bus del Sistema. ALU. Caché

- *Bus Sist.*
- **ALU** (Aritm. Log. Unit): en *Neumann*, el encargado de *Proc.* o hacer *Op.* enteras *Log.* (*NAND*) y *Aritm.* (suma, multipl.,) según **OpCod** o (Microordenes de Ud. *Control*). I: *Registr.* o Acumul. O: Registr. y Bit Estado (*CODE2*). También computa *Direc.* *Offset* como I. *Carg.* (*CODE2*). A. *Recurs.* (*Chomsky*).
 - **FPU** (Floating Point Unit): coproc. matem. de op. con \mathbb{R} (raiz,...). Suele venir Integr. en el *MicroProc.* Ver *Hist. A. Torres y Zuse IEEE 754*.

5.5 Unidad de Control

- Ud. **Control** (no confundir con *MC*): en *Neumann* es el corazón o *Autom.* que *Sincr.* los elementos.
El sist. *Secuenc. Ejec.* la *Instr.* que hay en el *IR* (Fig. 5.14) mandando *Signals* (**Microordenes**, OpCod a *ALU*) a los elemen. al ritmo del *Reloj* (marca los *Ciclo Maq.*). Vemos también el *PC* (direc. siguiente Instr.) y el *DeCod.*. Puede ser *Cableada* (*Hardw.ire*) o *MicroProgr.*. Ver *MicroC* (*Embeb.*), *Mem.*, *Gest. Mem.*, *MMU*, *C. Congest.*, *Multiplex.*, *NIC* (*Tarj. Red*), *TCP*, *Driver*, *Estr. C. Flujo*, *C. Calidad*

5.6 CODE2

- **CODE2** (*Comput. Didactico Elemental*): *Micro RISC* de *Palabra*=2 Byte=16 bits. Contrasta con *x86*. Fig. 5.14
 - *ALU*: solo 1 op. *Log.* (*NAND*).
 - *Mem.*: rango *Direc.*: hasta *FFFF*=64Kbi=16 bits (ver Instr. *Carg.* y los 3 *Bus Sist.*)

- *Puerto*: hay $256=8$ bits *Id.* de *Perif.* tanto para I ($IP00 - IPFF$) como para O ($OP00 - OPFF$) (ver Instr. *I/O*)
- *Registr.* General o *Fich.* R. o Banco R.: 16 ($r0, r1, \dots, rF$) cada uno Almac. Tam. *Palabra* ($FFFF$) como usados operandos/resultado/direc. Ej: Instr. *Carg.*¹ Ej: Instr. *I/O* de Puerto²
- *Registr.* Especificos:
 - *AR* o *MAR* (M. Addres/*Direc.* R.) y *DR* o *MDR* (Mem. Dat. R.): usados *Tmp* por Ud. *Control* para poder realizar las Op. R/W sobre *MP* anteriores. En otras CPU *MDR* es *MBR* un *Buffer* que va Almac. datos leiods. Ver *MMU*
 - *PC* (Program Counter): contiene siempre *Direc.* Mem. de siguiente I. (con respecto actual).
 - *IR* (Instr. R.): contiene siempre I. actual para que Ud. Control Read cuando desee en sus Ciclos de Ejec. Ver *Cuant.*
 - *SR* (R. Estado): BiEst.bles *Condic.* o Flag del resultado de última Op. *ALU*. Z (zero), S (sign), C (acarreo), V (*Overflow* o *Err.* o Excepc. *Manej.*). Ej: Instr. **Salto Condic.** o **GoTo**³
 - *SP* (Stack Pointer, *Punt. Pila Llam.*): contiene la última direc. de retorno (o contiene direc. que apunta a pila de direc. retorno??). Ej: Instr. *Llam. Subrut.+ Instr. RET.*⁴
 - R. Acumulador: no tiene (que es lo normal), pero si *RT* (R. *Tmp*) donde guarda uno de los 2 *Operandos* [209].
 - Otros: R. Constantes (almacena siempre 0,1 o π *Pi*).
 - Otros: *Interr.* son *Persist.*: se guarda su *Est.. CODE2* \neq Instr. de “*Interr.*” (ver)

5.7 Funcionamiento CPU

RESUMEN: Instr. APunt. por el Registr. *PC* (Progr. Counter) en *IR* (Instr. Regist.). Luego el DeCod. que posee traduce las Instr. binarias a señales en *ALU*, *Mem.* y *Puertos* (R/W del Bus).

- *HDD → RAM* (ini): cargar en RAM el programa a ejecut.
- *Addr(I₂) → PC* (ini): carga *PC* con direcc. de siguiente Instrucción.
- *I₁ → IR* (fetch): UC carga Instruc. a ejecutar en *IR*
- *Addr(I₃) → PC*: actualiza *PC* (por si *Interr.*)
- *Exe(IR)* (decode): UC interpreta la I.
- Según I. (execute): a) Load algo de mem. b) Write algo por Perif. c) Read de Perif. d) *ALU* o *FPU* op. e) Jump en programa. f) End programa. g) Volver a 3

¹o Almac. (R/W) $M(rD + v) \rightarrow rx / rx \rightarrow M(rD + v)$ con Form. $F3 = (codop, rx, v_a, v_b)$, consume 9 CPI Reloj
²o IN/OUT $IPv \rightarrow r_x / r_x \rightarrow OPv$ con Form. $F3 = (codop, rx, v_a, v_b)$

³(o Branch (bifurcación)): si Condic. de última op. *ALU* (*Z, S, ..*) se cumple ir a Intr. en Direc. del Registr. *rd* ($rd \rightarrow PC$) consumiendo 6/9 ciclos Reloj

⁴1) *Compil.* o *SO* decide zona para Pila de *Llam.* en *MP*. 2) Instr. *Subrut.* (o *CALL* o *Llam.* no *GoTo*) va apilando las Direc. de *Subrut.* para luego volver si Instr. *RETURN* (*Back Recurs.*) 3) *SP* apunta siempre a última direc. de retorno (pila Softw., aunque hay pilas Hardw.)

Ciclo de CODE2:

1. Inicia el *PC*
2. Ud. *Control* carga 1^a Instr.
3. Incrementa *PC*
4. Ud. *Control* desCod. la Instr. y manda señales para su Ejecuc.
5. Volver al pto 2.

5.8 Juego de instrucciones reducido, complejo y extendido

- **ISC** (*Instr. Set Computer*): conforman el L. *Ensambl.* que nos permite distinguir 2 tipos de *Micro*:
 - CISC (Complex): permite op. con operandos directamente de MP sin cargar a Reg-istr. En mayoria de PCs. Ej: *x86 Intel* y *AMD*
 - RISC (Reduced): tiene decodif. más sencilla (*Control*), las I. se ejecutan en un solo *Ciclo Reloj*, chips más baratos de fabricar, más *Efic. Energ. (Pot.)*, implementan *Pipeline*. Ej: *ARM (Movil, Tablets)*, *CODE2* o **RISC-V** [Berckley10] Arq. *Libre*
- Tipos de *Instr.*:
 - I. Básicas (solo *BCD* como *CODE2* o FPU simple, double, división por hardw. o softw.).
 - I. *Exten.* : los 2 más *Actual*. para *x86 Intel* o *AMD* son:
AVX (Advanced Vect. E., ver SIMD *Flynn*) y *AMX* (Advanced Matr. E., *Array*) para *Aprend. Autom.*. Atrás quedaron *MMX*, *SSE* o *3DNow*. Ver E. *C Harvard, Her., Texto E-BCD-IC, Exten. SQL/PSM (Proced. Almac.), ECD (Hardw. Red), Repet., Tranet, Ext. Red, Perif., Fich.. Ext4 Sist. Arch., Equiv. (Cl.-Struct), Mem. Virt. (MMU)*.
- Otros:
 - *Formato de Instr.*: suelen tener *Cod.* de *Op.* (OpCod o CodOp) y Operandos (en forma ex/implicitamente). Ej.: *CODE2 (codop,cnd, -, -)*
 - Mod. *Direc.*: *Traduc.*. Ej: indirecto, *Ind.exado*, *Offset*

5.9 Paralelismo Hardware y cache

- **Multiproc.** (no confundir con *Multitar.*): uso de más de una *CPU* en una *Comput.* (MIMD *Flynn*). Ej: *Plan. Proc.* con *Equil. Carg..* Comunicados con *Bus Sist. HyperTransport NUMA*.
- Multi**Core** o MultiNucleo: es varias *CPUs* en mismo chip o *Micro*. Fig. 5.14 Ver *Plan. Juanbe*. Ej: *Superescalar+MultiC=MIMD* Permite ejec. *MultiHilos* de un mismo o diferentes *Proc..* Hoy los *SO* saben evitar *Interfer.* de *Proc.*, sobre todo en lo relativo al *Recurso Mem. Compart.* (*Coher. NUMA SMP*). Políticas:

- *Master*: puede haber una CPU principal, Arq. ASimetr. de Intel
 - Mem. *Compart.*: UMA, SMP..
 - Hilos: *Plan.* Proc.
- *Cache*: L1, L2 y L3 Fig. 5.14

5.10 Ciclo máquina

- *Ciclo Máquina* Básico (no confundir con C. Reloj): cada una de las **Etapas** en las que se parte una *Instr.* 5.14 y son:
 - a) Fetch: *IF* (instruction fetch) (*Carg. Instr. de MP*).
 - b) Decode: *ID* (instruction decode) (en *Ud Control*).
 - c) Execute: *EX* (execute) (en *ALU*).
 - d) Store: *MEM* (update memory) (carga más resultado en *MP*).
 - *) Cache: *WB* (writeback to cache).

Fig. 5.14. A parte de las Etapas Ini (cargar en *MP* el programa) y End (parar). Usadas para medir el *IPC* y explotado en (*Pipeline Superescalar*).

5.11 Paralelismo a nivel de instrucción: pipeline, superescalar

- *IPC* (*Instr.* Por *Ciclo Reloj*, no confundir con *SO*) o $CPI = 1/IPC$: Metr.
- *Pipeline* (no confundir con *Pipe*): *Paral.* a nivel de *Instr.* dentro de un solo *Core*.
Lo normal es que sin Pipeline una I. ocupe varios ciclos ($CPI = 5$ Fig. 5.14)
Al partir las I. en las 5 Etapas *Ciclo Maq.* InDep. se llega a $CPI = 1$ ($CPI = 5$ Fig. 5.14), manteniendo ocupada cada Etapa de *CPU* y aprovecha sus *Recurso*.
Más típico en RISC
 - Thread-level parallelism (TLP) ($\subset \text{Pipeline} \supset \text{Task Parallelism}$): ejecuta I. de múltiples *Hilos* dentro de un chip procesador al mismo tiempo mientras que *Superescalar* significa ejecutar múltiples I. al mismo tiempo.
- *Superescalar* : *Paral.* a nivel de *Instr.* \iff $IPC > 1$ o $CPI < 1$! y en 1 *CPU* (similar DDR RAM). Hay 2 formas:
 - S.+*Pipeline*: se pueden ejec. 2 (o más) Etapas *Ciclo Maq.* distintas a la vez (2 *IF*, 2 *ID*,...) gracias a que tiene > 1 Linea Ejec., sobre todo en la Etapa *EX* (una I. usa *ALU* y otra Carga Mem.) Fig. 5.14, Hay *Recursos* duplicados con *Cache* L1i, L1d y 2 *IR* Fig. 5.14 y de esto se beneficia el Hyper-Threading <https://stackoverflow.com/questions/1656608/what-is-difference-between-superscaling-and-pipeline>
 - Hyper-Threading (no confundir con Multi*Hilo* ni *HyperTransport Bus*): es la *Impl. Intel* de la técnica Simultaneous MultiThreading (SMT, un tipo de Multi*Hilo* que expresa más el *Superescalar*, le mete más *Hilos*). En AMD se llama **Bulldozer**. La CPU simula 2 *CPUs Virt.* dentro de 1 CPU Física para hacer *Multitar.* o más bien multi*Hilo*. Dichas 2 CPUs aparecen *Transpar.* al SO que piensa que hay 2

realmente.

Lo minimo que requiere el SO es SMP (*Compart.*).

Apareció en Pentium 2002 *x86* y hoy lo llevan los *Intel i3,7,9*.

Unif. pues mantiene L. *Moore* como DDR RAM (*Benchmark*). Según nº *Cores* puede ser SISD, SIMD o MIMD (ver *Flynn*).

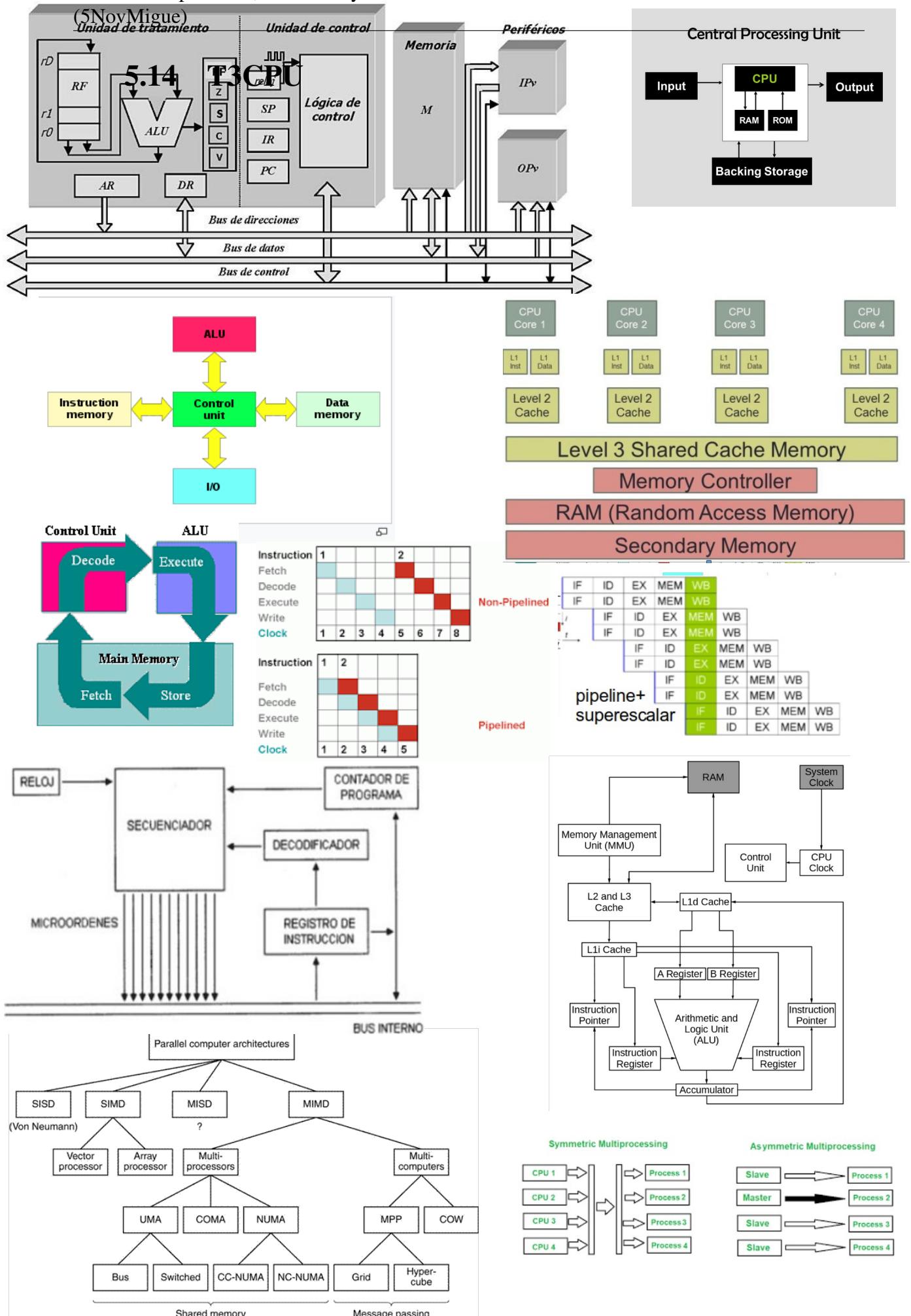
5.12 Paralelismo en nº proces: Flynn y memoria compartida

- *Flynn*: SISD, SIMD, MISD, MIMD
- *Compart.*: N/UMA, SMP

5.13 Microprocesadores populares

- Ver AMD e *Intel* y *Benchmark*

5 – +3. Componentes, estructura y funcionamiento de la Unidad Central de Proceso.



Chapter 6

*+4. Memoria interna. Tipos. Direccionamiento. Características y funciones. (24NovCarmen)

6.1 Introducción. Conclusión

- Ej: ¿ importa solo Capacidad en RAM? Sol. no también Vel. o A. Banda. (*Equil. Flujo, Conect.*)
- Hist. *Moore*: Valvulas, *Transistor*, ADRAM cada 4 años nueva DDR VRAM Anasagasti
- Futuro:
 - 1) Ultra *RAM* (*RAM+Masiv.*) unif. memoria y almacen.
 - 2) *MOSFET 3D Noel (HBM)* más Intensidad.

6.2 Arquitectura. Harvard modificada

- *Arq. Neumann*: Prog. Almac. en *MP* interna.
- *Harvard* Modificada (reflejada en *Cache L1*, alante)
- Fig. de Comunic. *CPU* (ALU, Registr., Control), *Bus Sist.*, *RAM/ROM*, *Perif.* (I/O, MS)

6.3 Memoria. Jerarquia

- *Mem.* o *Almac.* : componente que guarda *Inform.* (de *Dat.* o *Instr.=Palabra*) en celda (*Id.* con una *Direc.*) para que otros componentes (*CPU*, *Perif.*) puedan R/W. Distinguimos:
 - M. Interna o Central: pequeña y rápida, *Integr.* en carcasa, ligada a *ALU* y *Ud. Control.* Ej: *MP, Cache y Registr.*
 - M. Externa o Auxiliar o Secundaria o Drive: *Masiv.* y lenta. Ej: *MS (SSD o HDD), CD/DVD/BlueRay Disco, PenDrive (Flash), Petaca, Nube.*

Es el *Medio Comunc.* de los *Proc.* (*IPC*). Es *Unif.*. Ver *TAP*, *Recurso*, *Jerarq. Mem.*, *Gest. Mem.*, *Hamming*, *Mem. Compart.*, *Mem. Asoc.*, *Estr. Dat.*, *IA con M. Persist.*, *Var.*, *Punt.*, *Dinam.*, *Block M.*, *Almac. Distrib.*, *A. ADN (Biolog.)*

- ***Jerarq. Mem.*** (o *Capas*, o *Princ. Localidad*): *Jerarq.* donde cuanto más *Vel.* sea el *Nivel* menos *Almac.* y mas *Precio/Byte*. Fig. 6.15 la *Nube* podria ser el Nivel más alto. Cuidar *Coher.* para evitar conflictos.
 - *Fisic.:* *Registr.* *Cache*, *MP* e Internas, *MS* y Externas, *Nube*
 - *Log.:* ver *Fich.*

Explica porqué un *Naveg.* Web va lento y luego rápido tras una hibernación. Ver *Carg.*, *J. Bus (Chipset (Brigde North-South))*

6.4 Estructura de una memoria

- *Estr. Mem.:* ver Fig. 6.15
 - *Registr.:* contenido R o W reciente.
 - *DeCod.:* de *Direc.* col (CAS, ver abajo *Latencia*) o fil. (RAS).
 - *Circ. Control:* coordina R/W y Modo *Direc.* con un *Reloj* y lineas a elementos.
 - *Transduc.:* aporta *Energ.* para R/W.
 - *Matr. Array* (o soporte??): de celdillas de *Inform* que alberga 0 o 1.

Ej. pg. 190: construir una *RAM* de 64k palabras de 8bits cada una, *Compo.* en *Serie* y/o *Paral. Chips* 2141 (cada uno de 4K palabras de 1 bit). Sol:

6.5 Memoria: características

- Capacidad o Tamaño *Almac.:* *Metr. Inform. (Masiv.) KB..TB PB.* Se mejora con *Mem. Virt..* Ej: Rango *CODE2 FFFF*
- ***Volatil*** idad: si *Req.* uiere *Pot.* para *Preserv.* su contenido. Según t. distinguimos permanente, duradera, con refresco, V. y de lectura destructiva. Ver *RAM*, *ROM*, *CD/DVD*, *MS (HDD..)*, *Fich. (SAM)*, *M.2 (NVM)*
- A. *Banda* (o *Vel. Transm.*): depende de long. *Palabra* y Frec. *Reloj*. También del *Hardw.* (*Carg.*), *Latencia* y Mod. *Direc..*
 - *MTPS* (Millones de Transfer. *PS*): igual a Frec. *Reloj* (de *Controlador de Mem.*) o doble! (ver *DDR SDRAM*).
 - *MBPS*: *Palabra***MTPS*
 - T. *Acces.:* =1/*MTPS*, entre 2 op. de R (o W).
- ***Latencia*** (no confundir con Cap. *Canal*): retardo=delay=*Vel. propag.* que suele ser *Luz*.

- L. CAS (Column Address Strobe, Estroboscopia de la *Direc.* de Col. (de Luz Destellosa/Intermitente para ver obj. que gira como si estuviera parado)): en DRAM por vaciado del condensador, retraso en Ciclos de *Reloj* en Sincr. **DRAM** (o nanoseg. si Asincr. DRAM) entre el Comando READ (enviado por *CPU* o *MC*) a *Matr.* y momento que los Dat. están disponibles en los pines del módulo. Determina la *Vel.* de Acces. a Mem (**overclocking**).
- L. RAS (Row): tRCD (Row Address to Column Address Time), tRP (Row Precharge Time), tRC (Row Cycle Time=tRAS+tRP) y tRFC.
- A más tamaño más L. C. pero más *Reloj* ver Tab. DDR SDRAM.

Ver *Plan. Disco* (SCAN). *Red Superp.*, *Servic.*, *RT*, *T. Resp.*, *HyperTransport (Bus)*, *Cuello Botella*, *Cond. Carr.*, *Ping*.

6.6 Registros

- *Registr.*: R. General (*Fich. R.*) y R. Especif. (*PC* (contador), *IR* (intruc.), *SP* (pila) y los *SR* (estado))

6.7 Caché

- **Cache** o Intermedia o Antememoria: Mem. que se pone entre 2 Niveles *Jerarq. Mem.* por
 - a) *Prob. de Reus.* Dat o *Instr.* (=ctx *Markov*) y
 - b) actuar como *Buffer Vel.* (evitando *Cuello Botella* o T. Muertos).El sist. debe predecir cuales (a) y cuidar la *Coher.* (b)
 - C. Proces.: es *Volatil*. Antes menos pero actualmente 3 (Fig. 6.15 y Fig. de MultiHilo (*Pipeline*)):
L1: pega y separa L1 Dat. y L1 Instr. por *Harvard Modif.*).
L2: cada Core tiene una). El antiguo *Bus Sist.* Back/Front Side comunicaba L1 (*Integr.* a cada *Core*) con L2
L3: 1 para todo el *Micro* y contacta con *RAM*
 - C. *Disco*: ver *Arch. Dispos.*
 - Read: cuando un *Proc.* necesita un Dat. lo busca en L1, si no está en L2 .. hasta *RAM*. Luego lo copia en todas *L_i* para futuros Acces.
 - Write: debe mantener la *Coher.* (UMA o SMP *Compart.*). Comunicadas por *Bus Sist.* (Back Side Bus)
 - Arq: *MP (Harvard)*, *Registr.*
 - SO: *Persist.* *Maq.* *Virt.*, C. *Fich.*,
 - L.: Meomización (*Eval.* *Efic.*)
 - Red: C. de *Web*. C. *Móvil*,

Es *Unif.* por *Actual.* *Coher.* Ver *MMU*, *Pagin.* (TLB), *Proxy*, *Persist.*

- *Mem. Compart.*: en *Multiproc.* y *Cache UMA*, *NUMA*, *SMP* (cuidar *Coher.*, *Concurr.*, *Colis.*).

6.8 Memoria principal. RAM

- **MP** (Memoria Principal): tipo de *Mem.* Interna donde se lleva toda *Inform.* para *Proc..* Distinguimos *RAM* y *ROM*.
 - *Mem. Virt.*: Exten. con *MS*.
- **RAM** (RAM (Mem. Acces. Aleat.)): tipo de *MP*
 - a) De *Acces. Direct.* o *Aleat.* (*CPU* puede *Carg.* sabiendo *Direc.* ver Instr. *CODE2*).
 - b) de *R/W Volatil* y gran *Vel..*
Contiene los *Proc.* de *usr* (y *Servic.* del *SO*).
Distinguimos *SRAM* y *DRAM*. Ver *VRAM (GPU)*, *Kingston*
- **SRAM** (Static): basada en *BiEst..*
Comparada con *DRAM* no requiere **Refresco** y más *Vel.* (ver).
Hay *No/Volatile N/V-SRAM* según requieran *Pot.* (*Energ.*). Fig. 6.15
Ej: *Cache* pero también en Cámaras, *Rut.*, *Impr.*, o ciertas *MS*.
- **DRAM** (Dinam.): basada en Condensador+*MOSFET*.
Mas lenta que *SRAM* ya que *DesCarg.* un Condensador es más lento que cambiar un *BiEst.* y necesita **Refresco**.
Sin embargo tiene más *Almac.* (basta 1 solo *Transistor* mientras que *SRAM* 6).
Donde $1/0 = +/ - V_{cc}/2$ y la carga es $Q = -C * V_{cc}/2$ (*Metr.*).
Ej: en **MP DDR DSRAM, SSD o Flash**. Distinguimos:
 - ADRAM (ASincr.): del 60 al 97 que la reemplazó la SDRAM (*Hist. A.*). Sin *Reloj* solo cuando R/W activo?. Ej: en los *Intel Pentium Fast Page Mode (FPM)*
 - SDRAM (Sincr.): Circ. Secuenc. con *Reloj* desde 97 trabaja a misma *Vel.* que *CPU* (se sincroniza con el pero *Part.* entera, mas lento por *Latencia CAS ??*). La más Popular es la DDR.
 - Otras: *RAMBus* (no usada por cara, antes en *Nintendo2 Videojuego*) y *VRAM (GPU)*.
 - *Kingston* : mayor *Empresa* productora de *DRAMs* (para *Flash*).
 - *HBM 3D*.

6.9 DDR (SDRAM)

- DDR (Double Data Rate) SDRAM (Sincr. Dinam.): RESUMEN ej. abajo.
Transfer. en *Flanco* subida y en bajada gracias al *MC*, por lo tanto sus *MTPS* (Mega-Transfer Per Second) es doble de Frec. *Reloj* (ver *Bus Infinity Fabric*). Idea similar a *Superescalar*
Cada 4 años una nueva (*Hist. A.*) ver Tab. Sigue L. *Moore*.
DDR4 la más usada *Actual.* pero DDR5 la más moderna.¹ Es *Unif.* como *Superescalar*, mantiene L. *Moore*

¹la selección de *RAS* y *CAS* se hace exponencialmente a 2 en cada ciclo??

Caract. (año)	DDR1 (98)	DDR3 (07)	DDR5 (20)
Desensidad Modul. tipico	1-2GB	8GB (2700)	32 GB (6400)
Frec. Reloj (MHz)/MTPS(=2*FR)/A.	200/400/3200	800/1600/128000	2400/4800/38400
Banda(MB/s=2*8*FR)			
Volt (Pot.)	2.5	1.5	1.1
Latencia CAS (ciclos)			30

RESUMEN de Tabla: recordar la DDR5-4800: con 32 GB y 2400 Hz y cada 4 años una. $\Rightarrow MTPS = 2 \times 2400 = 4800$ (DDR) $\Rightarrow A. Banda = 8B \times 4800 = 38400$ MB/s

6.10 VRAM vs GDDR RAM

- VRAM (*Video*):
 - a) *Hist.* A. hasta finales de los 90 las *Tarj.* Garf. usaban Dual Ported VRAM (para *Render.*, actuando como *Buffer* y permitiendo a la vez a CPU W datos y a *Display R*).
 - b) Con la llegada de las SDRAM y su *Vel.* las Tarj. no necesitaron VRAM Fig. 6.15 (observar *MMU*, necesita UMA para *Coher.*).
 - c) **GDDR SDRAM** (Graphical SDRAM): ahora de nuevo las *Tarj.* gráficas tienen una VRAM llamada GDDR al rededor de *GPU* y de uso exclusivo para esta (Fig.). Son *Estand.* por el **JEDEC** y desde 2018, la más *Actual.* es la **GDDR6/X**: a 16 GbPS, poco Consumo *Energ.* a 1.35 V. La llevan GPU NVidea, Placa Madre y Videoconsolas (*Videojuego*) Playstation o XBox.
 - d) **HBM 3D** (abajo)
 - *IOMMU* (IOMMU, no confundir con MMIO TPerif.): es un ej. de MMU que conecta un *Perif.* (*DMA Bus*) con *MP Exten.* el Rango Direc del Perif (le da *Mem. Virt.*). Ej. AGP y *PCI Express* (*Intel* y *AMD*). tienen un IOMMU para las *GPUs*. Fig. 19.9 y 6.15 *Video RAM*
 - **HBM** (High Bandwidth Memory): SDRAM una mejora de esta en Espacio y Consumo. Con Met. *Acces. 3D*. usada por algunas *GPUs* y en *Red* de alto *Rend..* Noel dice que 3D aumenta Intensidad sin sobrecalentamiento (*Temp.*).
 - NAND Flash*: es la otra forma famosa de *Integr. 3D*.

6.11 Conectores RAM

- *Modul. RAM*: hay diferentes versiones *Estand.* por JEDEC según para PC o Portátiles. En consecuencia los *Conect.* Zócalos cambian (Fig. 6.15)
 - SIMM (single In-line Memory Module, no confundir con *Tarj.* SIM Movil ni Micro SD (*Flash*): es viejo, suele comunicar *Cache* con *Bus* de 32 bits.

- DIMM (double): Los hubo para SDR (168 contactos) pero hoy llegan al DDR4 (con 288 contactos). El SO (Small Outline) D. es para portátiles
- RAM DoubleCanal (no confundir con *Canal de I/O*):
Si ponemos 2 Modul. (gracias a un 2º MMU en el puente norte del *Chipset*) podemos aumentar al doble el ancho *Banda* de 64 a 128 bits accediendo a 2 Módulos diferentes. (*Paral.*)
El AMD Ryzen Threadripper (Tab. *Benchmark*) e Intel Core i9 Extreme Edition soportan QuadC. que es 4 veces más!
- Otras Caract. de RAM: Voltaje (ver Tab. DDR), Soporta ECC (*Err. Correction Code o CRC*)

6.12 Memoria principal: ROM

- **ROM** (Read Only Mem.): MP no Volatil y solo permite lectura. Se puede entender como un Circ. Combin. (Cod. seguido de DeCodif.) con I/O n/m bits. Ej uso: Subrut. POST de BIOS o EnCript. datos. Variantes (como Disco CD/DVD):
 - PROM (Programable ROM): se dan vírgenes y se programa una Unic. vez.
 - EPROM (Erasable PROM): Ej: BIOS. Se pueden programar varias veces con EM Ultravioleta.
 - EEPROM (Electricaly EPROM): Ej: evolucionan a Flash, pendrive USB. Como antes pero más barato al W con corrientes. No son Volatil.
 - DVD o CD-ROM (Disco Compacto): ISO 9660 define el Sist. Arch. Imag. Disco. Llevan CRC.

Ver RAM.

6.13 Direccionamiento: modos de acceso a SRAM, DRAM y CAM

- Modo Acces. MP (no confundir con Mod. Direc., ni Met. Acces. Fich. (ISAM)): por medio de pines, 1º Id. o activa la Direc. deseada y luego se R/W. Debe ser Efic.. Puede ser 2D (Matr.) (f, p) o 3D (Array HBM) (f, c, p) Fig. 6.15 donde cada tubo es una Palabra y la p seria el bit de esta??.
 - M. A. a SRAM (Estat.): por Palabras: **Direc. Cableado**, mecanismo inherente a su construcción, con Transduc. conectados a las Palabras. Puede ser 3D (HBM)??
 - M. A. a DRAM (Dinam.): por Block: con **Direc. CAS y RAS** (Fig. 6.15). Se Acces. Block (no a posic. individual, Arch. Dispos.) ⇒ la Direc. se parte en bloque+Offset ⇒ distingue capacidad bruta vs neta??
 - M. A. a CAM (Mem. Asoc. , Content-Addressable M., M. de Contenido Direc.): Mem. de Filos. Dat. Perd. o Consult.=Topos o Aprend. Autom. en que a diferencia de la tradicional le entra un Dat. o parte de este y devuelve las Direc. que cumplan

las *Condic.* *Busq.* de este. Usada en *Busq.* muy *Efic.* y *Vel.* como *Rut..* Puede ser *Cableado*, Tab. *Hash* o *NN Hopfield* o humano². Ver *Dic.*

- Otros: A. *Secuenc.* vs *Aleat.* (ver *Fich.* y *MS*). Mod. *Direc.* (como *Offset.* usados en *Segment.* y *Pagin.*)

6.14 Direccionamiento: modos

- Mod. *Direc.* (no confundir con Met. *Acces. Fich.* (ISAM)): dentro del *ISC* y el *Ensambl.*, es la forma de calcular la Direc. Fisic. *MP*, desde Dat. de *Registr.* y val. de *Operandos* de *Instr.*. Clasificación de Juan [Prito+Anasagasti, *Hist. A.*]:
 - Directos (o absoluto): $R0 + M(25A3h) \rightarrow R0$
 - Indirectos (o *Punt.*): simple $R0 + M(R7) \rightarrow RO$ doble y triple. Ej: *INode Recurs..* Ver Yushchenko
 - *Ind.exado.* (o *Offset* o Relativo o Desplazado o Compensando): $R0 + M(R7 + 0Bh) \rightarrow RO$ Ej: *Punt.+O.* Ej. I. Cargar *CODE2*. Ej. El O. de una *Signal* es su DC. Ver *Pagin.*, *Acces.* Dir. Rel. (*SAM*). .
 - M/PMIO (ver)
 - Otros: implícito, inmediato... de forma más general puede ser un *Alg. Busq.* como en *Estr. Dat. Arbol* o *Hash* (no solo basado en la D. si no en la misma Inform).

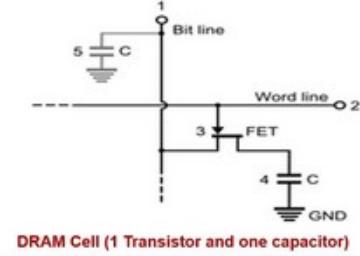
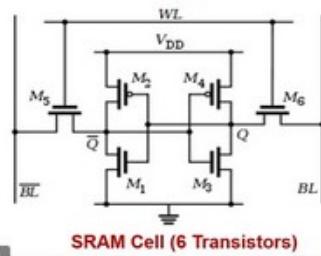
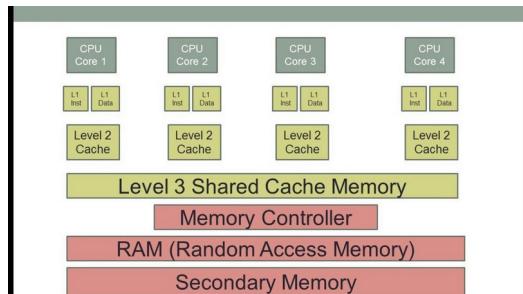
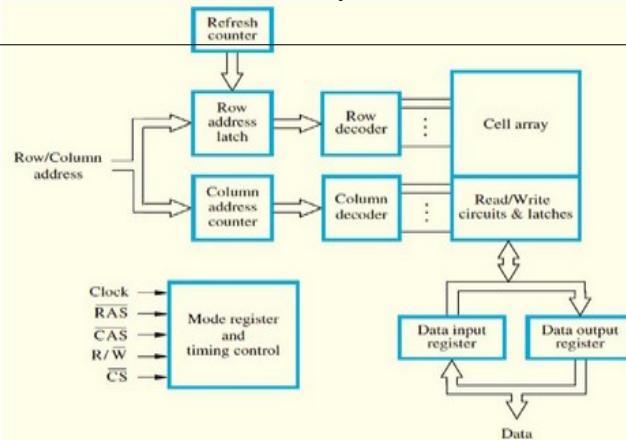
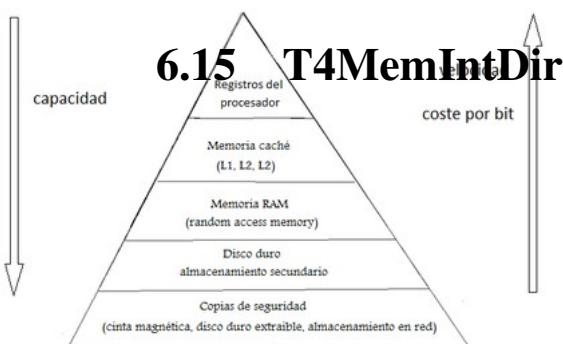
Es *Unif.* por *Topos*. Ver *Punt.*, *Pas.* por *Ref.*, *Enumer.*, *ISC*, O. *Signal*, *x86*, *Traduc.*, *Id.*, *Acces.*, D. *IP* o *MAC*, *Duplex*, *Mem.* *Virt.*, *MMU*, *Etiq.*, *Key*, *NAT*

- *Direc.: Id.* (en forma de *Tupla*) Asoc. a una Inform. o *Palabra* para *Busq.* en *Mem.* (o un *Esp.*). Suele ser un *Num.* **Hexadec.** como *Punt.* (en Bytes *Binar.*). Ej: *Var.* vs *Punt.* o *Rut. Dir.*
 - *Esp.* o Rango D.: = Capacidad Mem. Rel. con *Palabra* y *Bus.* Ej: *CODE2*, *Mem.* *Virt.*, *MMIO (I/O)*
- Mod. *Direc.:* Clasificación de Pedro Anasagasti:
 - *Impl.ícito:* Direc. dato I. o Especificada en cód. de op. Ej: *Registr.* o *Pila*
 - Inmediato: Valor del dato contenido en propia Instr.
 - Directo: La Instr. contiene la direc.
 - * Directo Absoluto: Ej: Registro o por Memoria
 - * Directo *Ind.exado* o Relativo a Registros: se añade un *Offset.* Ej: *PC*, *SP*, Base-Desplazamiento, Reg. General, Índice, Autoindexado (de(increm))
 - Indirecto: el campo del op. contiene direc. de mem. en la que está la direc. efectiva dell operando. Ej: *Punt.* y sobre todos los anteriores

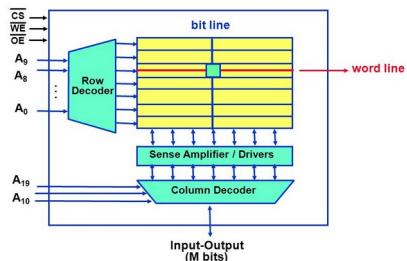
²no recordamos nombre de persona nº 343 conocida si no una que es físico y empieza por N (sol. Newton)

6 – *+4. Memoria interna. Tipos. Direccionamiento. Características y funciones.

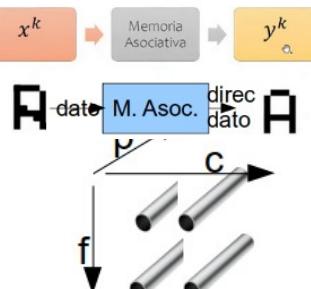
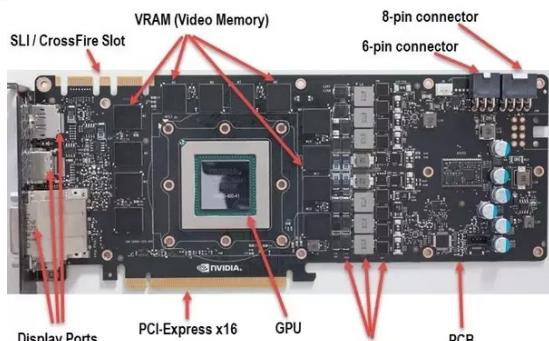
(24 Nov Carmen) UNIDAD DEL COMPUTADOR



SRAM Memory Architecture



Independent system RAM and Unified memory video RAM



Comparison of DDR Generations:

DDR Version	DDR1	DDR2	DDR3	DDR 4	DDR5
Released date	2000	2003	2007	2012	Under Progress
Operating voltage	2.5V	1.8V	1.5V	1.2V	1.1V
Prefetch buffer size	2	4	8	8	16
Chip densities	128Mb-1Gb	128Mb-4Gb	512Mb-8Gb	2Gb-16Gb	8Gb-64Gb
Data rate (MT/s)	200-400	400-800	800-2133	1600-3200	3200-6400
Bank groups	0	0	0	4	8
Termination/ODT	On board	ODT added	Nominal, Dynamic Modes	Park Modes	Nominal Wr/Rd

Comparison of DDR Generations

Comparison of memory modules for portable/mobile PCs (SO-DIMM)

Chapter 7

5. Microprocesadores. Estructura. Tipos. Comunicación con el exterior.

7.1 Introducción. Conclusión

- Ej. motivador: ¿Que es el Model. Prod. Tick-Tock de Intel? PC componentes compatibilidad. (*Flujo OK*). *Ing. Inv.* (ver).
- Hist.: *Moore* crea *Intel*, 4004 de *Intel* 71 el 1o . Acuerdo 82 x86, *Bus Sist.* a Front Side Bus por QPI
- Futuro: *Cuant.*

7.2 Arquitectura del ordenador

- *Neumann, Harvard* Modif.

7.3 Estructura estandar de CPU

- Fig. *CPU* normal y con MultiHilo (*Pipeline*) con L1i, L1d, y doble *IR*
 - *Registr.* general y específico (*AR, SP....*)
 - *Bus Sist.* (ver abajo *HyperTransport*).
 - *Ud. Control:* microordenes
 - *ALU:* OpCod.
 - *Cache:* L1i,L1d,L2 y L3

7.4 Microprocesador: características

- **Micro** Proc. (no confundir con MicroControlador (*Embeb.*)): *Impl. Equiv.* de una (o más) *CPU*, donde la *Ud. Control, ALU* y están *Integr.* en un solo *Chip* (de silicio con millones

de *Transistores* (ver)).

Suele incluir otros Proc. como FPU o de I/O (como iGPU o sonido). *Ejec. Instr.* en L. Maq. *Ensambl.* (Op. básicas aritm., log. y de carga o I/O)

- MicroProc. *Vect.*: ver *Supercomput.* o AVX (*ISC Exten.*)

7.5 Características Físicas

- Caract. Básicas (*Benchmark*):
- Rel. con *Bus Sist.*
 - Longitud *Palabra* (Word Size): nº bits *TAP* en *Paral.* en *CPU* relacionado con *Bus Dat.* Actual. 32 y 64 (también 128) bits. Ver *Simbol.*, *Registr.* *CODE2*, *IEEE 754*, *Texto*.
 - Bus *Direc.*: limita rango *Direc.* máx (ver *Carg.* *CODE2*)
 - Otros: *Bus Sist.* (*HyperTransport*)
- Rel. con *Vel.*:
 - Frec. *Reloj* o CPS (Ciclos PS de R., no confundir con $\lceil IPC \rceil > 1$! (*Pipeline*)): de Ud. *Control* para *Sincr.* los elementos.
Internacional (o del *MicroProc* en *GHz*) vs Externa (o de *Placa* con *Bus* y *RAM* en *MHz*). Ver *Secuenc.*, *Tmp*, *Cod.*, *AutoR.* (*Manchester*, [Self-clocking Signal])
 - MIPS/MFLOPS/MVPS?? (Millones *Instr.*/Flotantes/*Vect.* Per Second)=*Pot.* Comput. ver *IPC Pipeline*
 - Otras: *Latencia CAS* y de Acces. a *Perif.*
- Rel. con *Fisic.*:
 - *Pot.*: TDP (Thermal Output), VRM, consumo *Energ.* 150W
 - *Tecn.. Integr.*: *Moore* 10nm, Fig. 7.12. Trans. 3D
 - *Conect.*: Zócalo (*Socket*) 1500 pines
- *Encapsul. Micro* (Embalaje o Empaquetado): protección puesta a la oblea de silicio para conectarle Pines *Puertos* y protegerle (mecánicos, *EM*, *Temp.-refrigeración*). Se *Conect.* al Zocalo.
- *Puerto/Pin*: nº *Id.* de los Pin *I/O* de un *Micro* que sirve de *Interf.* o *Conect.* con otros *Dispos.*

7.6 Características Softw.

- Rel. con *Instr.*:
 - C/RISC + Juego de I. Básico (*Op.* *BCD*, Flotantes, Div. Hardw. o Softws.) + I. Extend. (AMX para *Aprend. Autom.*).
 - Otras: *Tabla Vect.* de *Interr.* (IVT, *Manej.*), *Interr.* para *Perif.*

- Rel. con *TAP*
 - Si *iGPU* o *Chipset* (si P. Norte) incluidos
 - N° *Caches* (L1, L2, L3) *Registr.*, *iMC*. Otros: *DMA*
 - N° *Cores* vs N° *Hilos*: Hyper-Threading de *Intel (Superescalar)*. *FPU (ALU)*
 - P/MMIO
- Según Paral. Instr.: *Pipeline* y *Superescalar*
- Según Paral. *Multiproc.*: *Flynn* (Proc. Vect.)

7.7 Comunicación con exterior: I/O

- *I/O*
- *Comunic.* Exter.: *Micro* (o *RAM*) → Pin Puerto → Socket → Bus Sist. → *RAM* (o *Micro*) o *Placa* → Bus I/O → *Perif.*

7.8 Comunicación con exterior: bus del sistema

- *Bus Sist.*: (forma antigua) de comunic. los 3 elementos principales *CPU*, *MP* y *Perif.* Se parte en:
 - a) B. *Direc.* (de *MP* o *Puertos*),
 - b) *Dat.* (long. *Palabra*) y
 - c) *Control* (para *Sincr.*). *Actual.* solo en *Micro Embbed.*, en M. normales fué hasta 80s, luego reemplazado por el Front Side Bus y actual InfinityFabric Comun. *MultiCores* Fig. 7.12:
 - RESUMEN *Hist.* A.: UniBus (DEC69) → FrontSideBus (FSB, Intel90), EV6-Alpha (AMD) → *HyperTransport* (AMD01, HTX3.1-2014 a 3GT/s, sirve como reemplazo del FSB para comunicar con *Chip Puente Norte*, también sirve para comun. *Multiproc.* con NUMA y para comun. *Rut.* vs *Ethernet*), QuickPath (Intel08) → InfinityFabric (AMD16), Ultra Path (Intel17 pa Tick-tock Skylake, 10 GT/s=80GB/s)
 - Otros: *Canal* (lineas eléctricas) que *Transm. Dat.* entre *Compo..*, normalmente en *Paral.* (L. *Palabra*) pero también en *Serie* (ver *Conect.*).
 - Otros: *Jerarq. Chipset*: *Bridge North (PCI)*, *South*

Ver *Red Bus*, *Cuello Botella*, *RAMBus*, *Instr. Carg. CODE2*.

- *DEC UniBus*: del famoso PDP-11 (1969 *Hist. A.*, *Thompson* lo usó): 1º con MMIO y permitió DMA fácil. Se usó como *Bus del Sist.* (comun. CPU y MP) y como *Bus de Perif.* (CPU y Perif.). Como era habitual, los Perif. no tenían un Bus para ellos (comparten Bus Sist. con CPU *Red Bus*).¹ Ver *Guerra Protoc.* DECNet. MODULA (*Modul.*), Radia Perlman (*Recubr. STP*)

¹DEC Popular Empresa comput. 57-98 → Compaq (producia PCs para IBM) → HP. Los Perif. como el Bus del Sist. (que comunic. MP y CPU) se unifican (son el mismo, por ello Uni). Esto permite hacer DMA y MMIO fácilmente. Flexible y económico pero con el problema de supeditar Vel. a Dispos. más lento??

- **Front/Back Side Bus (B/FSB)**: antiguo de Intel y AMD 1990-2005, reemplazado por Intel QuickPath Interconnect y *HyperTransport* para comunic. CPU con *Chip P. Norte*. El BSB comunica *Cache L1 (Integr. a CPU)* con L2 (fuera) Fig. 7.12. Reemplazado por *HyperTransport*.
- ***HyperTransport*** (no confundir con Hyper-threading (*Superescalar*)): antiguo de AMD 2005-2015, reemplazó al FSB y reemplazado por Infinity Fabric. QPI (QuickPath Interconnect) de *Intel* Fig. 7.12) y DMI (Direct Media Interface) para *Intel*.
Pero ambos van hacia **Infiniband** e Infinity Fabric (ver). Son: *E2E* bidireccional *Serie/Paral.* de baja *Latencia* y Alto Ancho *Banda*. Asumen que el Micro tiene un *iMC* integrado, obligando así a los *Multiproc..* a usar una Arq. NUMA (Mem. *Compart.*).
- **Infinity Fabric**: nuevo AMD desde 2016, mejora *HyperTransport* para conectar *GPU* y *CPU* la cual desde la Arq. Zen'18 trabaja a la misma F. *Reloj* que la *DRAM* por lo que usar una RAM más *Vel.* hace todo el Bus más rápido.
- **DMA** (ver UniBus).

7.9 Historia de los Micros

- *Hist. A.:* Intel 4004'71 4bits, Intel 8088'79 (*x86 Acuerdo 82*), Intel Pentium'93 (antes 286'82 16bits, 386'85 32 bits, y 486), PowerPC'92 (*IBM+Apple+Motorola en Macintosh*)
- *Arq. x86* : familia de *Micro CISC* y *Estand.* de facto sobre Arq. MIPS, SPARC (no confundir con *Arq. ANSI/SPARC..*) Contrasta con *CODE2*.
Actual. son *Harvard Modif.* (*Cache L1* datos y *L1 instr.*)+*Neumann* (*L2, L3..* mezcla datos e instr.) y llevan Hyper-Threading (*MultiHilo*) (ver Fig.).
Acuerdo del 82 *Hist. A.:* <https://parceladigital.com/articulo/cuando-los-microprocesadores-fueron-hechos-en-esta-fecha/>
Intel subcontrató a AMD para que hiciera micros Intel 8088 e Intel 8086 (porque no daba a basto para IBM). *IBM* crea sus PCs y quiere Micros x86 pero si dejaba a AMD hacerlos también, por ello todos hoy. *Segment.* es *Legacy Pagin.* lo Actual. Llevan: Tabla *Vect.* de *Interr.* (IVT), *Instr.* TSL (*Excl.*), PMIO
- Ing. *Inv.:* necesario hoy porque *Intel* es no *Libre*

7.10 Fabricantes Actuales

- Otros *Populares*: Qualcom, TSMC, *IBM*, MediaTek, Samsung. pero los principales para PC y Portatil son *Intel*, *AMD* y *ARM (Movil)*.
EPAC'2022: 1º *Micro* de la U. *Europ..*
- ***Intel*** : Empresa del i3, 5, 7 y 9 (hasta 12^a gener.). RESUMEN: en Tick-Tock tuvimos Skylake (2017 del UltraPath a 14nm), hoy RaptorLake (Sept 2022) que compite con AMD Ryzen 7000 a 10 nm
 - Tick-tock: modelo de *Prod.* de I.
Tick (aumentar *Integr.*, reducir tamaño)
Tock (nueva *Arq.*).
Por L. *Moore* se espera cada año un Tick-Tock pero *Actual.* no lo cumple por el costo de *Integr..* Parece Alfa-Beta de *Vers.*

- * i5f (no tiene Gráfica y necesita *GPU*) vs i5 (si la tiene pero se *Temp.* más y es más caro). Otros: *Microcontr.* 8051. Ver Tab. *Benchmark*.
- * **IMPORTANTE Arq.** Proc. *ASimetrik. Multiproc.*: unos mejores que otros *Master Slave*. Ver *Equil., SO*.
- * *Hist. A.: Moore68.* Sus *Microproc.* compiten con AMD (Advanced Micro Devices, su CEO es una mujer Lisa Su) y menos con ARM (ver RISC).
- * Llevan Hyper-Threading: ver *Superescalar*

Ver *x86*.

- Otros: Atom, Celeron, Serie X, Xeon (*Servid.*, Quark SoC y Pentium).
- **AMD: Ryzen 3,5, 7 y 9.**
 - Otros: serie Athlon, APU (mi Portatil, integran *GPU*), EPYC (*Servid.*), Ryzen PRO, Ryzen Threadripper
- *Mac Apple*: los M1: M1 Pro, M1 Max, M1 ultra
- *Movil*:
 - ARM: para *Movil* y *Embeb.*
 - Otros: Bionic A13/14/15, Snapdragon 8/70/88,..
- *MicroControl*: ver *Embeb.* 8051, *Arduino*

7.10.1 Benchmark de micro actual

- ***Benchmark*** (Challenge): *Test Estand.* del *Rend.* (*Vel.*) de un *Compo.* (*Micro, Mem., GPU, Traf. Red*). Ej: Tab. *Compo..* Ver *QoS, Superescalar, Actual*.

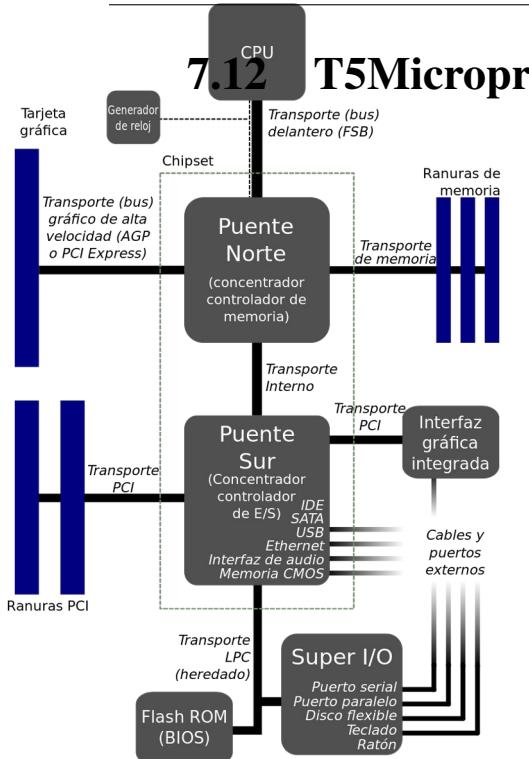
<i>Compo. Popular (Precio)</i>	<i>Benchmark MIPS/Reloj</i>	<i>Caract.</i>
AMD Ryzen Threadripper 3990X (7000 Euros)	2 mill!/4.35GHz	64 Cores/ QuadCanal RAM
<i>Intel Core i5-9400F 2.9GHz</i> (150 Euros) Ver <i>TConect.</i>	/2.9 Ghz	6 Cores, iGPU

PC Componentes te *Config.* compatibilidad. Ej Actual: - Socket: AM5 Socket (AMD5 Ryzen 9 7a gen.) o Intel Socket 1700 i9 12 gen. - RAM: DDR5

7.11 Cuantica

- *Comunic. Cuant.* : RESUMEN no *Intercep.* por entrelazamiento de espines de fotones por *Fibra*, record chino y Nobel2022 Alain Aspect.

- QSDC: Quantum Secure Direct Communication (Comunic. Directa Segura Cuant.): para Distrib. Claves Cuant. Record Chino 100km a 1bps (muy lento). Fotones por fibra entrelazados spin. No *Intercep.* por Observar=Altera. Si 30 km audio móvil speed
 - **Entrelazamiento** y superposición son la base.
 - NobelFisca2022 Alain Aspect, Anton Zeilinger *Hist. R.* por fotones (o electrones con spins) entrelazados (entangled) y mostrar la violación de desigualdades de Bell. Fig. 7.12
 - *Intercep.*: es evitada.
 - Agujero de Gusano=Entrelazamiento: 1^a simulac. C. de A. G. [ElPais30Nov22]
- *Comput.* C.: trabaja a *Temp.* de 0º K. *Actual.* va por 400 (IBM [Pais9Nov22]) qbits pero para Apl. de *Opt.* necesita al menos 1 millón. Se ve imposible por ahora ir a más qbits salvo nuevo breakthrough, aunque [Youtube20NovShortConGafas] dice que con campos magnéticos nuevos podríamos llegar al millón.
 - Qbit: programar fotón al gusto, para que 60% de las veces arriba y 40% abajo (*Binar. NAría*)
 - Alg. Shor: amenaza RSA
 - **T. Coher.** t. que permanece un qbit estable (sin colapsar) permitiendo R/W (como *IR CODE2*)
 - Superposición C.: es como una *F*. que *rePresenta* muchos valores diferentes (Comb. *Lin.* de estados) y que se *Eval./Proyec.* al medirla.
 - Futuro: Qualcom se va asociar con *Europ.*, hay que doblar la producción de Semiconductores en un mundo en el que todo estará conectado.

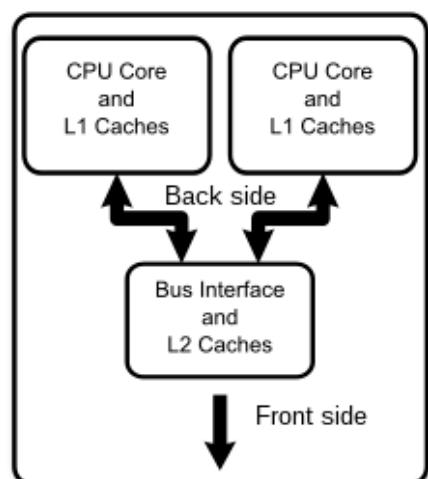
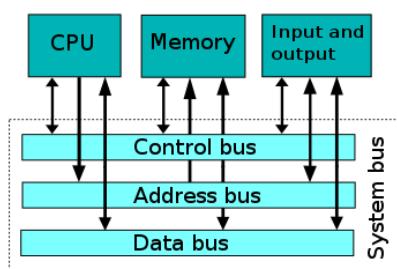


MOSFET scaling (process nodes)

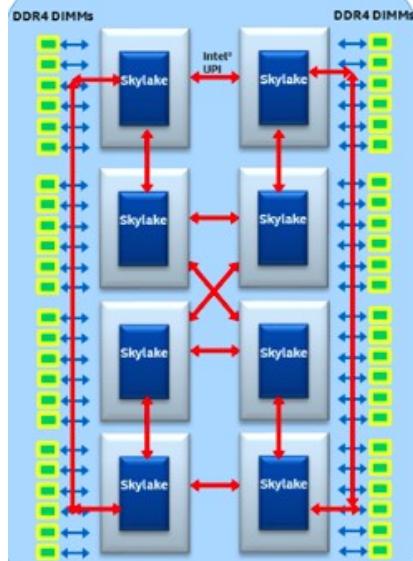
- 10 µm - 1971
- 6 µm - 1974
- 3 µm - 1977
- 1.5 µm - 1981
- 1 µm - 1984
- 800 nm - 1987
- 600 nm - 1990
- 350 nm - 1993
- 250 nm - 1996
- 180 nm - 1999
- 130 nm - 2001
- 90 nm - 2003
- 65 nm - 2005
- 45 nm - 2007
- 32 nm - 2009
- 22 nm - 2012
- 14 nm - 2014
- 10 nm - 2016
- 7 nm - 2018
- 5 nm - 2020
- 3 nm - 2022

Future

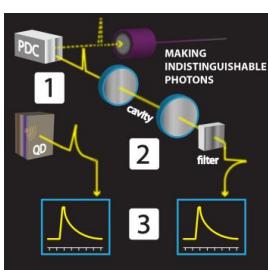
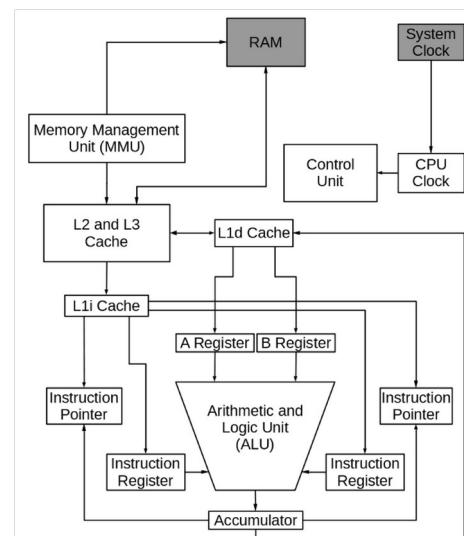
2 nm ~ 2024



Intel Ultra Path 17



AMD Infinity Fabric 16



Chapter 8

*6. Sistemas de almacenamiento externo. Tipos. Características y funcionamiento.

8.1 Introducción. Conclusión

- Ej: ¿Preserv. Backup.?
- Ej: Ley Moore: Metr. Binar. Kib = $1024 = 2^{10}$. ¿Límite Fisic. Holografico Cuant.?
- Hist. A.: Tarj. Perforadas (MS) Disco vinilo (Brazo se repite hoy). Floppy Tecn. Winchester (HDD de IBM 1973) VHS Flash Hologr. EEPROM
- Actual: Video a demanda: Netflix.
- Futuro:
 - 1) Disco congelados o ADN Biolog. molecular (Almac. MS)
 - 2) UltraRAM (RAM+Masiv.) y MOSFET 3D Noel (HBM) (ver MP, HBM)

8.2 Computadora von Neumann. Jerarquia de memorias

- Neumann Harvard Modif.: reflajado en Cache L1.
- MP (donde se lleva info para Proc.). MS: situar en Fig. 8.10
- Jerarq. Mem.:

8.3 Almacenamiento externo y Memoria secundaria y

- **MS** (Mem. Secundaria): Exten. de MP (para Mem. Virt.).
Mem. Externa: Dispos. para Almac. Masiv. y no Volatil.
En ambos Precio Euros/Byte < RAM pero menos Vel.. Fig. 8.10.

- Ej: *HDD, SSD o Nube*.
- Ej: *Tarj. Perforadas (Punched Card)*: *Hist. A. Req.*uiere un Lector de T. Primero usado en Telares de Joseph Marie Jacquard 1800 (creo que mecánico). Luego *BCD-IC* de *IBM (Texto)*. Luego Millipede Memory de *IBM 2005* no triunfó.

Ver *Estr. Dat. Ext. Disponib. RAID (Redund.)*, *Preserv., Almac. ADN (Biolog.)*, *Kingston. Moore*.

- *Estr. Fisic.*: como *Perif.*
 - IMPORTANTE: No confundir:
 - a) Soporte (DVD) con
 - b) *Ud. Perif.* o Sist. de R/W (Lector-DVD): con *Cabecera Transduc.*, *Log. Control..*
 - *Tecn. soporte*:
 - a) *EM Magenética*: *HDD*
 - b) *Luz Óptica*: *DVD*.
 - c) *Flash*: *SSD*
 - d) *Nube*
- Caract. generales:
 - Capacidad: *Esp. Cuota* GB o TB.
 - *Vel.*: *A. Banda*
 - Ubicación: interna, externa o *Nube*.
 - Mod. Transf. (*DMA* o *PIO*), *Cache Disco*
 - Acces. y Arch. Dispos.:
 - a) *Aleat.*: $< t_{max}$ (*HDD, SSD*).
 - b) *Secuenc.* (*DVD*),
 - c) Dispos. *Block=Buffer* (*SSD, HDD*)
 - *Reus.*: n° W (*Trim*).
 - *Softw.*: adicional para R/W (ej. *DVD* si para W, *Flash* no).
 - *Conect.*: abajo

8.4 Soportes magnéticos

- *Disco* Magnetico *EM* (D. Optico abajo): como vinilo: sustrato magnético+*Brazo+Cabecera*. Coord. polares (sector+pista, ver *HDD*).
 - Suelen ser *Reus.* y de *Acces. Aleat..*
 - *Cache* de D.: *Buffer* o porción de *RAM* o *MS* asociada un *Disco* en caso de *Reus..* Ej: *Arch. Dispos. Block*.
 - *Plan.* de D.: *Brazo SCAN*

Ver *Block Mem. (Arch. Dispos.)*, *CD-ROM*, *Floppy D.* *Cache D. (Arch. Dispos.)*, *Imag. Disco, Part. D., Cuota D. .*

- **HDD** (Hard Disk Drive, *Ud. Disco Duro*): sustrato duro para no deformarse al girar a gran *Vel.* (**7000 rpm**). Cerrado al vacío. Hay internos y removibles (*Hot*).
 - *Fisic. Brazo-Cabecera (Signal de R/W)*.
Soporte (material magnetizable+plástico)
 - *Tecn.* Winchester: varios *Discos* formando un cilindro. Resultó en un ahorro en complejidad y se volvió *Estand.* Nombre en código (referente al fusil) de los HDD de IBM 3340 1973 *Hist. A.*¹
 - *Log.:* **cord. polares** (r, ϕ) Fig. 8.10
Pista r : círculo, numeradas de fuera a dentro **como tocadiscos**, apilados forman un cilindro (ver Winchester abajo)
Sector ϕ : trozo de pista, juntos forman un *Cluster* (que es la *Ud.* de *Fich.*) ver Ej. Disco Opt. abajo. *Block*: forma de transf. no bit a bit.
 - *Arch. Dispos.* de *Block*: luego tiene *Buffer* o *Cache* de *Disco*=porción de *RAM* (abajo).
 - a) Cap.: GB o TB .
b) A. *Banda*: MB/s (*T. Resp., Brazo*)
c) *Energ.* consumida:
 - *DMA*, *Block Arch. Dispos.*

Ver *Kingston*

- *Disco Floppy* o *Flexible* o *Diskette*: Tenían pestaña de *Permiso W. Magnético*, obsoleto *Hist. A.*, el 3+1/2 con 1.44 MB era *Popular* en 1987.
- Cinta Magnet.: De *Acces. Secuenc.*.. VHS o casete, obsoleta *Hist. A.*, plástico-flexible+oxido-magnetizable. Baratas pero lentas. solo para *Backups* o estudios de *Audio*.

8.5 Ópticos y magneto-ópticos

- *Dispos. ópticos o Luz* de R/W: *Brazo+Laser* (como *Disco*).
 - **Cod. Land-Pit** [Eight-to-Fourteen Modul.]: Cod. *Lin.*
 - a) **Microagujeros en metal-reflector+plástico-policarbonato.**
 - b) Basada en *Flanco, Reloj* y Diferencial e *Interfer.* destructiva de luz. Fig. 8.10
 - a) Soporte extraible (*Hot* transporte fácil).
b) *Precio/Byte* bajo.
c) Más *Integr.* que magnéticos como floppy (su capa protege de campos *EM* y corrosión).
 - Otros: *Acces. Aleat.*
Requieren *Softw.* adicional para R/W
- Tipos de D. Opt.: todos igual tamaño *Estand..* Se diferencian en:
 - a) densidad de *Integr.*: en *Cod. Land-Pit* Fig y dist. entre pistas. 8.10
 - b) película protectora.

¹Permitía a las cabezas aterrizar y despegar de la superficie del disco cuando éste paraba y arrancaba

- CD (*Disco Compacto*):
 - a) (Capac., Vel.) = (700MB, 150Kbs).
 - b) *Sist. Arch. ISO 9660*
 - c) Llevan *CRC*. Variantes como *ROM*:
 - a) *CD-ROM*: grabados en fábricas para lanzar grandes tiradas de *Public.* (*Despl.*) como encicloped., manuales, SO.
 - b) *CD-R (WORM)*: W Once Read Many
 - c) *CD-RW (Erasable)*: RW many
- DVD (Digital Versatile Disc, *Disco Versatil Digit.*): (4.7GB o 17GB) *Sist. Arch. UDF* (Universal Disk Format). Hay de 2 caras. Para *Videoclubs*.
 - a) *DVD-ROM*
 - b) *DVD-R/+R....* (como CDs 1W-MR)
- Blu-Ray: (25 o 50 GB) si 1 o 2 capas. **Luz ultravioleta**. Para ultraHD *Video*.
- **Hologr.** apic Versatile Disc (**HVD**): mejora Blue-Ray y HD-DVD (4TB). Gabor 1971 descubrió la H. *Hist. A.* Ver *Fisic. Cuant.* (Intro MS), 3D, *Luz*
- QUITAR: Magneto-opticos: mixtos donde R con *Luz* y W con *EM*. Principal ventaja es que no se degradan con la reescritura. Desventaja: coste, lentes y la apoyan pocas *Empresas*

Ej. 8.6, en un CD de 650MB de 74min y cada sector es de 2KB/sect (*Block*) ¿cuantos sect. se leen por seg?: Sol. $75\text{sect}/\text{seg}$ pq $N\text{Sec} = 650\text{K}/2 = 325\text{Ksect}$. $\text{Sec}/\text{seg} = 325\text{Ksec}/(74 * 60)\text{seg} = 73\text{sect}/\text{seg}$

8.6 Soportes electrónicos o flash

- **Flash** :
 - a) Evol. de las EEPROM (programables y borrables electricamente) basadas en *Transistores MOSFET* o *NAND* o *NOR* Fig. 8.10. *Hist. A..*
 - b) Mas concretamente tiene **Floating Gates MOSFET por eso dos metales** Fig. 8.10, donde hay 2 transit. uno de *Control* y otro de *Sens.* donde si carga <50% en Puerta Flotante es un bit 0.
 - Ventajas: bajo *Precio*, *Portab.*, no requiere *Softw.* adicional de R/W.
 - Ej: Pendrive, *Tarj. Mem.* o *SSD* (ver abajo)
 - *NAND 3D* F. junto a *HBM* son ej. famosos de Integr. 3D.
- **SSD** (Solid State Drive, *Ud. Est. Solido*):
 - a) normalmente basadas en *Tecn. Flash NAND* (más modernas).
 - b) pero también en *DRAM* (más *Vel.* pero necesitan bateria interna para *Persist.*),
 - c) híbridas *HDD* (grande)+*SSD* (rápido). Reemplazando a *HDD*.
 - V: arranque (*Rest.*) *Vel.*, resistir golpes (\nexists *Brazo*), menos tamaño y mas *Efic. Energ.* (útil para *Movil*).
 - D: *Trim*, menor t. vida, *Precio*, *Fall.* inesperados.

Tienen *Cache*. Usa *Conect.* SATA o M.2 con NMVe (*Conect.* abajo) Ver *Comando Trim* (*Plan. Disco, Asign. Sist. Arch.*), *Plan. SSD (Brazo)*, *Kingston*.

- *Tarj. Mem.*: para *Dispos.* como *Tablet, Movil, cámaras..* Requieren un lector de tarj. Distinguimos:
 - a) Micro/Mini SD (no confundir con SIMM RAM o SIM Movil): *Segur. Digit.*, de cámara.
 - b) Otros: CF (Compact Flash), SDHC, SDX, XD-Picture, Memory Stick..
- *Disco Flash* (o Mem. USB o Pen Drives o Ud. Lápiz): 128MB a 2TB, del *USB* toman *Energ.*

Ej. de *Banda* basado en *Latencia DRAM* (ver)

8.7 Conectores

- *Conect.:*
 - a) **Soporte removable** (no confundir con *Hot*): (*USB* como *DVD Disco, Portab.*).
 - b) *Ud. R/W* unidos (como *DVD*).
- **M.2** : especificación (no un slot concreto) para *Dispos. Flash, Wifi o Bluetooth* reemplazando al mSATA (que usa *PCIe*). Se *Conect.* en un *PCIe 3.0 x4* (aunque también *SATA3.0*, y *USB 3.0*). Ver Fig.
 - **NVMe** (Non Volatil Mem. Express): *Interf.* para que mediante *PCIe* se pueda acceder a *Flash* como *SSD*. **Con M.2 es el más Vel.**, más que M.2 SATA (600MB/s vs 3500MB/s, 6 veces más) (ver Fig.).²
 - *SATA* (más para *HDD*) y *M.2 NVMe* (para *SSD*).
 - **U.2**: quiso reemplazar *SATA* sin éxito, usado en *SSD*, comparado con *M.2* permite desconexión *Hot Mezcla SATA+PCIe*. Fig. 8.10.³.
- Otros: *Wifi* o *Bluetooth*

8.8 Almacenamiento distribuido. RAID. Nube

- *Almac. Distrib.:* SAN, NAS y DAS
- **Redund.** RAID (Redund. Array of Independent Disks, Grup./Matr. Redund. de Discos inDepend.): conj. de *Ud.s MS* para Alta *Disponib.* y *Preserv.*. El SO los ve *Transpar.* como un solo Volumen (*E:*). Si *Fall.* un Disco el conj. se recupera de *Perd.* y sigue f.
 - Usado en *Servid.* Fig. 21.13
 - Barato y fácil de implementar

²<https://www.kingston.com/es/blog/pc-performance/two-types-m2-vs-ssd>. <https://hardzone.es/reportajes/comparativas/ssd-sata-vs-nvme-comparativa>. Antiguamente llamado NGFF

³<https://hardzone.es/reportajes/que-es/interfaz-u-2-ssd/> antes SFF-8639

- Aumenta *Rend.* en: a) *Vel.* de R/W. b) *Fiabilidad, Segur.*
- *Niveles* (de *Paral.??*): RAID 0/1/5....
- *Array Disco*: 2 tipos *Almac. Distrib.* (NAS o SAN) y *Almac. Virt.* (*VFS*).

Es *Unif..* Ver *Entropia, Topol.* Red Doble Anillo, *Compr., Err., CRC, Arbol, FN (BD), Backup (Preserv.)*.

- *Nube*: ver *Almac., Transpar. de Recursos, Protoc.* NAS, Dropbox, iCloud.. *BD Nube*

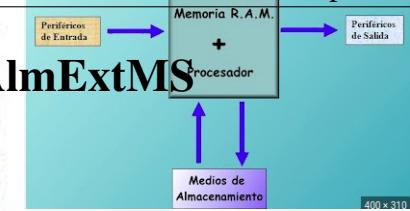
8.9 Otros

- *Preserv. Digit.*: de *Mem..* ej. *Migr.* dat. para que estén siempre *Disponib.* (aunque cambie los lectores de *MS*). Ver *Persist., Volatil, P. Docum.. Instal.*
 - *Backup (Copia Segur.)*: con *Comando tar* podemos indicar: Full, Incremental y Differential (completa, incremental y diferencial). También hay simple, diaria o en la *Nube*.

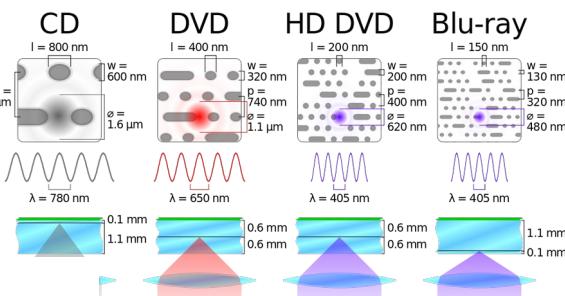
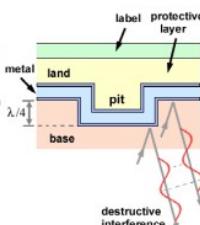
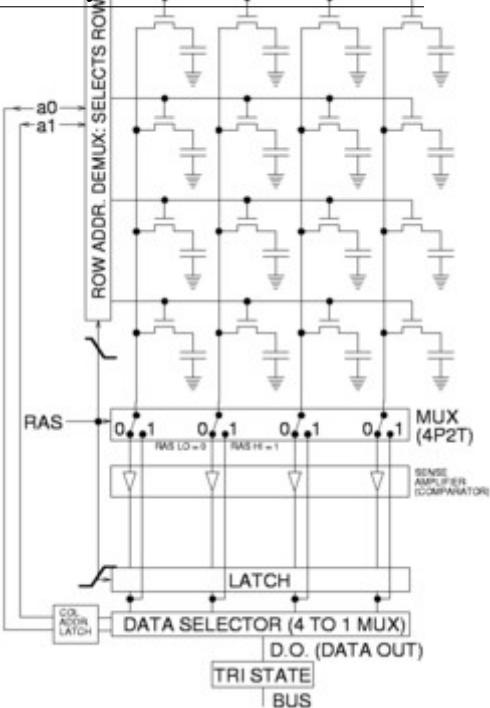
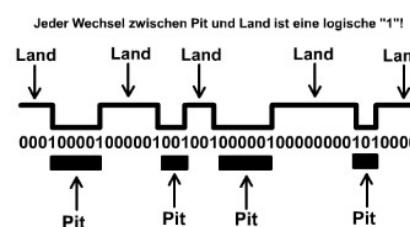
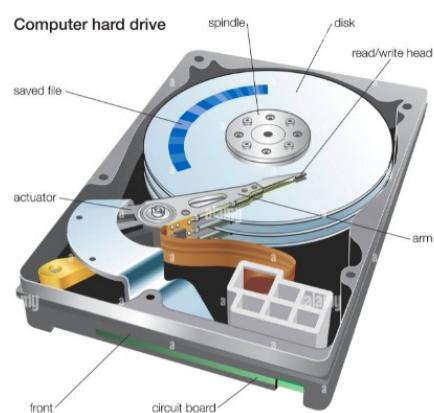
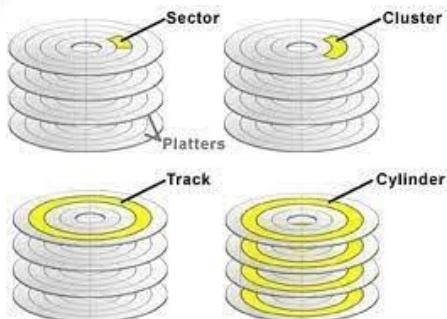
Ver *Redund.* (RAID)

SSD vs HDD

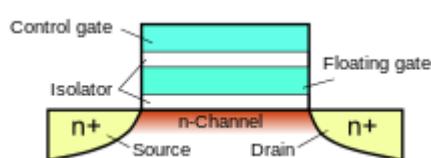
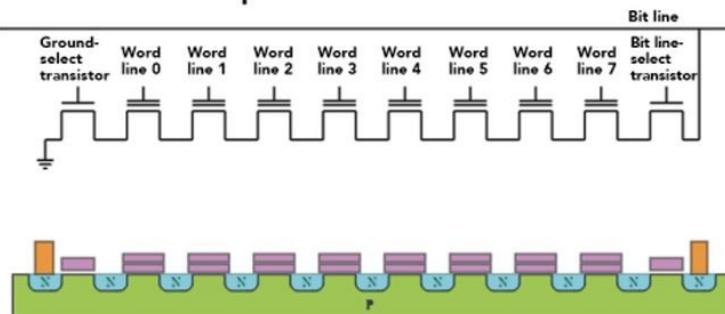
8.6. Sistemas de almacenamiento externo. Tipos. Características y funcionamiento.



8.10 T6AlmExtMS



Simple NAND flash structure

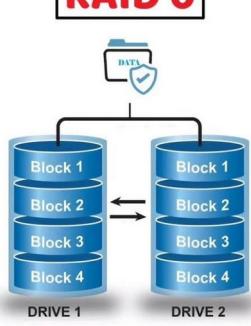


NAND flash memory structure is characterized by serial-linked groups of memory

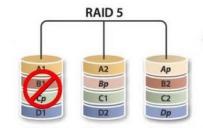
What is RAID?

- RAID 0**
- RAID 1**
- RAID 5**
- RAID 10**

RAID 0



Data is not duplicated



RAID level 5 – Striping with parity



Chapter 9

*+7. Dispositivos periféricos de entrada/salida. Características y funcionamiento. (21EneEva)

9.1 Introducción. Conclusión

- Ej-motivador: ¿Como *Compo.* OK? Sol. *Flujo Opt.* (*Display*)
- Hist.: *Serie 232* → *PS/2* → *USB*. *UniBus* de *DEC* (1º *MMIO*)
- Futuro: *MiraCast (Wifi)* *Chat-GPT Robot* o *IoT. USB* para todo (*WebUSB IoT*).
Tablet Display flexible, *3D (Hologr:amas, Spielberg)*, Gafas Realidad *Virt.*

9.2 Arquitectura de computadoras. Placa madre

- *Neumann Harvard* (separa Dat. e Instr. actual. en *Cache L1*)
- *CPU, Bus, MP, Placa.*
- *Chipset* (Norte, Sur) *Perif.* y Fig. 9.12 (*PCI, Super I/O*).

9.3 Periférico. Estructura

- *Dispos.* (“a secas”) o *Ud.:* cualquier *Compo.* o *Sist. Hardw.* (máquina, eléctrónica, biológica..) capaz de realizar un *TAP*. Ej: *Estac. Trabajo, Hardw. Red, Robot, Perif.* (=D. *I/O*). *Admin. D.* (*devmgmt.msc*). Ver *Arch. Dispos.. Web Responsive, Supercomput..*
- *Perif.* : cualquier *Dispos.* de *I/O* o *Interf. Comput..* Permite *Comunic.ar CPU* con el mundo *Fisic.* mediante los *Puertos*. Es una *Exten. Hardw.* del Sist. *Inform..* es *Hot* (des/Conect. de *CPU* sin problemas) (la fuente Aliment. no lo sería). Ver *IoT, Esper. Activa+Interr.*

9.4 Estructura hardware de los periféricos

- *Estr. Fisic.:*
 - 3 partes ([Juan]): pensar en teclado o pantalla *TAP* (ver *TI/O*):
 - a) *Transduc.: Actuad.* (motor) o *I. Sens.* (acústica, óptica, magnética,...)
 - b) *Buffer: Registr.*
 - c) *Controlador de I/O*: (ver abajo, no confundir con *Driver*) *Chip* en a) Perif. b) Tarj.
 - c) Placa
 - (d) *Conec.*: Comunic. con Comput. 3 del *Bus* (abajo)
 - (e) *Driver*: Progr.
 - IMPORTANTE: Perif. vs Soporte: (*MS*)
 - ELIMINAR: 2 partes ([Prieto]): Mecánica + *Electr.* [Javier] Mecanica+Control. (Interf. con Comput. y dividido en Interf. P. + Registr. dedicados + Interf. Bus-CPU??).
 - 3 Lineas: de *Autom.* (casi *Bus Sist.*)
 - a) *Est.*,
 - b) *Control* y
 - c) *Dat.* (según Est. leido enviar unas *Signal* de Control).
- *Controlador de Perif.* o de *I/O* (no confundir con *Driver*): es un *Chip*.
 - Ej. Super *I/O*: conectado a *SouthBridge* (Fig. Wiki) para dispos. lentes (Floppy-Disk, Ratón, Teclado, *Puerto Serie*). 1º Tarj. luego *Chip* de *Placa madre*, originado en 1980 por *IBM* (*Hist. SO Hist. A.*)
 - *Proc. I/O*: puede *Impl.* en a) Dispos. propio, b) *Tarj. Controladora* (insertada en zócalo). c) Integr. en *Placa Base* Puedes comprarlo por 20 Euros.
 - *Control de I/O*: *Est.* y Datos. Ej. *DMA*, Procesad. E/S, *Tarj. Control*, con *Puertos E/S. Monit.* por Control Softw. de Dispos.

9.5 Tipos

Tipos *Perif.* según:

- *Flujo Info:*
 - I (teclado, webcam, escaner).
 - O (pantalla, *Impr.*, altavoz jack y óptico!).
 - *I/O* (HD, Pantalla Tactil, Lector CD). Pueden ser *Recursos en Red* (*Impr.*, fax). Ver *Socket, DMA*
- *Cooper.:* de uso:
 - *Excl.usivo*: un *Proc.* cada vez, el SO debe evitar *InterBloq..*
 - *Compart.:* *Concurr.* cuidada por SO. Ej: *Disco* y *Spooling*.
 - *Virt.:* P. de uso *Excl.* pero parece *Compart.* por *Spooling*. Ver *Arch. Dispos..*
- Esp.-Tmp-Vel.:

- Esp.:
 - a) Local (*Impr.* local)
 - b) *Red* (*Impr.* red)
- Tmp *Reloj*:
 - a) Sincr.: Ej: DDR RAM??
 - b) ASincr.:
- Vel.: *Chip* Norte (rápidos), Sur (lentos)
- Otros:
 - Uso: TAP
 - a) *Ud. Comunic..*
 - b) Ud. de Almac.: (*Exten. MP*) Fig. 9.12)
 - *Conect.:*
 - a) Serie/Paralel.
 - b) *Hot*
 - Modo Op.:
 - a) On-line (comunic. continua y en *RT*, ej: *Monit.*).
 - b) Off-line (largos T. Espera e Independ. del Comput. Ej: *Impr.*).
 - *Arch. Dispos.:* c,b (*Acces. por Buffer*), s (ver)
- *Interr.* (abajo):
 - a) PIO: o sondeo (*Esper.* Activa) PMIO o MMIO
 - b) IIO: o interr. el Dispo. dice “listo”
 - c) DMA

9.6 Direcciónamiento de periféricos: PIO

- *Direc.*iónamiento: *Id.* de *Arch. Dispos.* comunicar con CPU por *Puertos+Registr.* Generales. Ej: *CODE2* 256.
- **PIO (I/O control. por Progr.):** RESUMEN: Modo *Direc.* donde cada *Petic.* de Transac. a *MP* entre Proc. y Perif. requiere una *Instr.* del *ISC* de CPU (que la para) a diferencia del *DMA*. Puede ser de dos tipos (*TPerif.*):
 - **PMIO** (port-mapped I/O): Instr. especial pa R/W Puerto Ej: IN/OUT de *x86* y *CODE2, Intel??*
 - MMIO (memory-mapped I/O): misma Instr. de R/W Mem. Ej: *Embeb.* *UniBus*
 - Implica *Esper.* Activa (o sondeo, la CPU comprueba periódicamente el Registr. *Est.* del *Controlador* de I/O).
 - Ej: PATA y *Conect.* antiguos (PS/2 del Ratón, Joystick y *Legacy* MIDI).
- **MIO (Map. I/O):** Estr. según *Direc.*iónamiento **PIO** (cada transf. requiere Instr. CPU que la para *TI/O*) distinguimos:
 - **PMIO (Puerto MIO):** poseen una *Instr.* especial para R/W en Puertos dando más complejidad. Va en *Micros* con long. *Palabra* <=16 bits (para no desperdiciar *Mem.*).

- * Ej: Instr. *IN/OUT* de *CODE2* y *x86* .
- **MMIO** (*Mem.* MIO, no confundir con *IOMMU*): usa el mismo *Esp. Direc.* (\Rightarrow mismo *Bus*) para Mem. que para *Puertos*. reservandose una parte para estos últimos (8000 – 80FF). Da menos complejidad, menos lógica interna (*Filos.* RISC) y permite *Reus.* las mismas Instr. de Mod. *Direc.* que Mem para Puertos.
 - * UniBus: de *DEC*, 1º con MMIO, servía como Bus Sist. y Bus Perif.
 - * Ej: en *Embeb.* (la baja complejidad da más *Vel.*, menos *Energ.*).
 - * Ej: QUITAR: *Intel* (pero no AMD) puede ser PMIO algo¹

9.7 Puertos o conectores

- **Conect.** (o *Puerto*): *Interf. Hardw.* entre *Dispos.* y *CPU (Placa)*. Suele tener varios pines, distinguirse Macho y Hembra (*Jack*).
 - IMPORTANTE: *Compo. Opt.* de Conect.: ver Ej. *Display*.
 - a) *Serie*: *USB*, *Ethernet*, *HDMI*. Llevan *CRC*.
 - b) *Paral.*: salvo en *RAM*, IMPORTANTE **reemplazado por Serie** por bajo *Precio* de Circ. *Integr.* Ej: PATA vs SATA.
 - **Hot** (en caliente, no confundir con *Plug and Play Win (DMA)*): *Perif.* o *Compo.* o *Dispos.* *Swipeable* o *Plugeable* o removable sin tener que apagarlo. Ej: U.2 (SATA), *USB HDD* externo. Lo contrario es **Cold** como *CPU* o *RAM*. Ver *Cod.* One-Hot *Vect.* (*NLP*), HotSpot (*WAP*), *Robot*.

Es *Unif.* por Compo. Opt.=*Equil. Flujo* (ver *Display*), *Conect. BD*

- *Benchmark* de Conect. Actual: PC Componentes Apire3 A315 43 R4VC AMD Ryzen5 (6 nucleos) 15 pulgadas 8GB RAM, 512 GB M.2,
 - *RAM*: DDR6 (*Modul.* SIMM)
 - *GPU*: PCIe x16 (Fig.)
 - *SSD M.2 NVMe* (que usa *PCIe x4*) o *SATA* (lento)
 - *Wifi y Bluetooth*: por *M.2* muy rápido
 - *Display*: DisplayPort
- Otros *Conect.*:
 - a) *Video*: *Display*, *HDMI*
 - b) *Audio*: *Jack*
 - Pines *Arduino*: *Analog.* (abajo A1-A5) y *Digit.* (arriba 0-13, con PWM).
 - Red:
 - a) *Cable*: *RJ*, *SC*, *BNC*, *Patch*, *Tarj. Red*, *Transcep.* (Balun), *WAP*.
 - b) *Wireless*: *Bluetooth*, *Wifi*

¹Con *Palabra* 32 o 64 bits donde no se nota desperdicio *Mem.* en reserva para Puertos. Es más PMIO límita a 1 *Registr.* En *Intel EAX, AXyAL* son los únicos Registr. a los que se pueden mover datos, y un valor inmediato de tamaño byte en la instrucción o un valor en el registro *DX* determina qué puerto es el origen o el destino de la transf.

- **PCI** (*Perif. Compo. InterConect.*): *Bus* y forma más común de *Conect. Tarj. Controladora* a *Placa madre*.
 - **PCI** Normal: lento, conect. al *Brigde South*.
 - **PCIe** (Express): muy Vel., en **Chip Norte**, después de *RAM* 2º *Bus* más Vel.. Variantes Fig. 9.12 según pines:
 - x4 para *M.2 NVMe*: mejora para que PCIe acceda directo a *SSD (M.2)*
 - x16 (para *GPU* o *SATA* (ver *M.2*))
 - IOMMU: donde se comunica con *GPU*. Reemplaza al AGP para *Video* (marrón).

Ver *Chip P. Sur. Thunderbolt (Display)*.

- **P/SATA** (*Paral./Serie AT Attachment*): Distinguimos:
 - **SATA**: en **Chip Sur**. Para *HDD* o *SSD* o *Disco CD* a *Placa*. Lleva *CRC*. Reemplazó al *PATA*. Distinguimos varios: eSATA, mSATA...
 - a) **mSATA** (mini): de *Placa* base a *MS*, pinchas y bajas, **sustituyendose por M.2**.
 - b) **eSATAp** (Power): =eSATA+USB, permite conectar a eSATA un *USB!* Fig.
 - **SAS** (Serial Attached SCSI): compatible y misma forma que SATA Fig. pero mas *Precio*².
 - **PATA** o **IDE** (*Integr:ated Drive Electronics*): típico *Bus* de 16/32 cables grises, antiguo y reemplazado por SATA.
- **USB** (*Universal Serie Bus*, no confundir con *Serie 232* o *UniBus (MMIO)*): en **Chip Sur**, **permite Hot**. Visto como *Arch. Dispos. Serie* (*ttyACM0* o *COM6*). Lleva *CRC*.
 - USB-c: es el futuro, reemplazará a todos los *Conect.* incluso de *Red*. Ej: para datos y cargar pantallas,..
 - **Firewire** (no confundir con *Firewall*): reemplazado por *USB 2.0. IEEE 1394*, Fig. 8.10 parece *USB+HDMI*. Para conectar en *RT* camaras en *VA Industrial* y *MS*, Tiene versiones en *Ethernet CAT5* y *Fibra*.
 - **Serie 232** (RS-232): para *E2E* directo (sin *Comut.* paq.). Reemplazo por PS/2 y *USB* (ratón y teclado, Super *I/O*) Envía datos en *Serie* muy lento (< 20 kbps y < 15 m) comparado con *Ethernet* y reemplazado en *Perif.* por *USB* . *Estand.* por EIA60 (*Hist. R.*). Puede trabajar en modo *Sincr.* o *ASincr.*(en este último cada trama o Byte se usa 2 bits de parada/stops). Su canal 2º más lento y para ACK (*AcRe*). Ej: *Impr.* documentos a 15m o *Modem*. Ver *LAN viejas??*

Ver *Socket WebUSB (IoT)*

9.8 Dispositivos de entrada: características y funcionamiento

- Teclado: *Transduc. Pulsamecánica* → *Signal Puls. ASCII??.*
 - Según Forma: T. *Win* (104 teclas **QWERTY vs Dvorak más egonómico** usa 77 por ciento la fila media, la más rápida y cómoda,), T. *Multimedia* (teclas de Acces. dir. a *Internet*), T. Ergonómico (más grande más teclas), T. Flexible (se dobla), T. *Proyec. Infrarrojos* o *Laser* en una superficie)

²más para *Servid.* y alto *Rend..* Puede R/W cualquier *Ud.* *SATA*, el *Recipr.* no es posible.

- Según Construcción: T. Contacto dos tipos Fig. 9.12: a) Mecánico (émbolo+placas cuadradas de cto) y b) Membrana (lámina+cúpula). T. Sin Cto: más caros, por cambio de Volt. (Pantalla Táctil??).
- Según Conect.: PS/2, Bluetooth o USB
- Manej. del Cursor o Puntero: para Click o Menu
 - Ratón: *Mov.* → *Coord.*: R. Mecánico (con bola), R. Óptico (*Sens.-Emisor Laser*, como S. Dist. *Robot*). R. Wireless (en *RF* mejor que *Infrarrojo* por obstáculos). Otros: TouchPad (el del Portatil), Trackpoint (el rojo en mitad)
 - Joystick: hoy también de O porque vibran. Para *Videojuegos* y Dis. Graf.
 - Lápiz Optico (ver *Tablet Digital*)
- Lector de Inform.:
 - L. *Tarj.* Perforadas (ver)
 - L. Magnet.: para *Tarj.* crédito y Libretas de Ahorro (*Sucursales*)
 - L. Opt.: a) *Cod.* Barras, b) Cod. QR, c) OCR (Opt. Character Recon.) para talones *Sucursal, Test, Loterias..*)
 - Escáner: *Digit.* papel. Caract.: a) *Resol.* (en *dpi*) b) *Vel.* *pg/min*, c) Profundidad Color: 2^{nbits}
 - Webcam: poca *Resol.* Pixel y FPS
 - *Tablet Digit.alizadora* (ver)
 - Micrófono (ver *Jack*)

9.9 Dispositivos de salida: características y funcionamiento

- *Display*, *Monit.* o Pantalla: visualizar *Inform..* param. *Resol.* y Frec.
 - *Resol.* : Dots Per Inch $dpi = ppp = \text{Pixel/pulgada}^2$ de una *Imag.. Actual.* 4K significa casi 4000 píxeles horizontal, el más común es de 3840 x 2160 píxeles. Ver *Cuantiz.*
 - a) Tamaño: Portatil 16 inch, PC 22 inch. Relación de aspecto 4:3=1.33, 16:10=1.6=Φ
b) Frec. Barrido (no confundir con FPS que es la salida de *GPU*): 60 Hz o 75 Hz (puede ser entrelaz. 1º lineas pares luego impares en *CRT*???)
c) N° Colores: junto a Resol. es la calidad principal 8-bit (256 colors), 16-bit (65K), and 24-bit (16M).
 - RESUMEN: consumo de *Pot.* 100 W. Las LCD antiguas se retroiluminan con halógenas y las modernas con LED blancos (por eso al poner negro se ve algo, el LCD decide el color). Las LED antiguas se retroiluminan con LED blancos y las modernas OLED no tienen retroiluminación es todo RGB LEDs (por eso el negro se ve negro puro, ej. en *Movil*).
 - * CRT: antigua. Usa Rayos Catódicos, ver Lápiz Opt., *Impr..*

- * LCD (Liquid Cristal Display): media. Usa retroiluminación de lámpara o de LED. Usa Luz polarizada. LCD-TFT mejoradas por LCD-IPS³
- * LED (Light Emisor Diode): moderna, Efic. Ecolog.. Usa diodos. (OLED, QLED Cuant., AMOLED).
- * Flexibles: para *Tablet Movil*.
- a) **HDMI** (High-Definition Multimedia Interface): para *Displays*. Usa *Cod. Manchester* 8b/10b Serie. Creado por conj. Empresas Philips, Toshiba,... . Siendo reemplazado por MiraCast que usa *Wifi Direct* (ver).
- //b) *DisplayPort*: Conect. rápido (ej. Compo. Opt.). Luego Thunderbolt⁴.

¿Compo. Conect. Opt.? Equil. de Flujos en Tarj. video: de nada sirve Monit. 4k si es Tarj. FullHD. <https://www.xataka.com/basics/vga-dvi-hdmi-displayport>

IMPORTANTE: Elegir Compo. OK Por Vel.:

-) Otros: RCA (3 cables), Euroconector. Firewire (USB), S-Video (parece Cannon Jack)
 - a) VGA (FullHD)
 - b) DVI (3,96 Gbit/s,)
 - c) HDMI (10,2 o 18 Gbit/s 4k a 60Hz)
 - d) Thunderbolt 3 (10 Gbit/s, 5K o Dual 4K a 60Hz, por USB-c)
 - e) DisplayPort (32,4 Gbit/s 8K de 7.680 x 4.320 a 60 Hz compatible con USB-c)

Ej. 8.1 puntos dpi en pantalla de 14 pulgadas

- **Impr.** : Inform. → Papel. Perif. y Recurso Red (*Samba*).

- a) *Vel.* (por caract. o lineas/min antiguas y lentas *Actual.* por pg./min.),
b) *Resol.*, c) Tipo del Soporte (papel A4, A3???)
- *Tecn.:* de peor a mejor Fig. 9.12 compara Tinta vs Laser.
 - a) *Matr.* o de Agujas: como máq. escribir (martillo con matriz de agujas+cinta tinta), por caracteres (lenta?? y mala calidad)
 - b) Inyec. **Tinta**: 1º campo EM ioniza gotas, 2º campo las desvia a su posic (como CRT), 3º Reus. de tinta sobrante (por receptor con bomba). Por caract. o pg. y buena calidad.
 - c) **Laser**: 1º Laser por donde pasa, carga negat. puntos de un cilindro 2º partículas del toner (positivas) se pegan a puntos del cilindro 3º las partículas se pegan al papel al girar el cilindro. Por pg. (Vel. y buena calidad).
- Otras:
 - a) Térmicas: no requiere toner pero si un papel especial. Por Lineas y mala calidad.
 - b) Inyección Cera: la tinta es cera que se derrite
 - c) Por Sublimación de tinta: pasa de sólido a gas, para papel especial. Calidad muy buena.
- I. 3D (.stl)

³IPS, también conocido como In-Plane Switching, es un tipo de *Tecn.* de visualización de monitores y pantallas. Más específicamente, un panel IPS es un tipo de LCD TFT LCD (o LCD de *Matr.* activa). Fig. 9.12 AMOLED vs LCD-IPS

⁴de Intel y Mac para remplazar al *HDMI* en TV, combina *PCI*

- Plotter: dibujan líneas (Atractor Lorentz) moviendo un Lápiz con motores paso a paso precisos. Son de pocos colores (monocr. o 4) y pueden mover un rollo. En pasado para *Dis.* CAD en Arq. o Delineac. en papel grandes (pues hacían líneas más Vel. que Impr.)
- Otros:
 - * SO: *Servid.* basado en *Spooling*: crea colas *Buffer* de tareas: El SO crea un *Dir.* especial y un *Daemon*, donde el Proc. vuelve el Fich. *Tmp* y se pone hacer otras tareas. (*Filos.* *Petic.*)
 - * R: CUPS (Common Unix Printing System): <http://localhost:631/> 631 es el Puerto TCP por defecto de CUPS. El Comando *lpadmin* lo abre.

Ver *Informe*.

- *RFId.* (ver Ropa)

9.10 Dispositivos de E/S

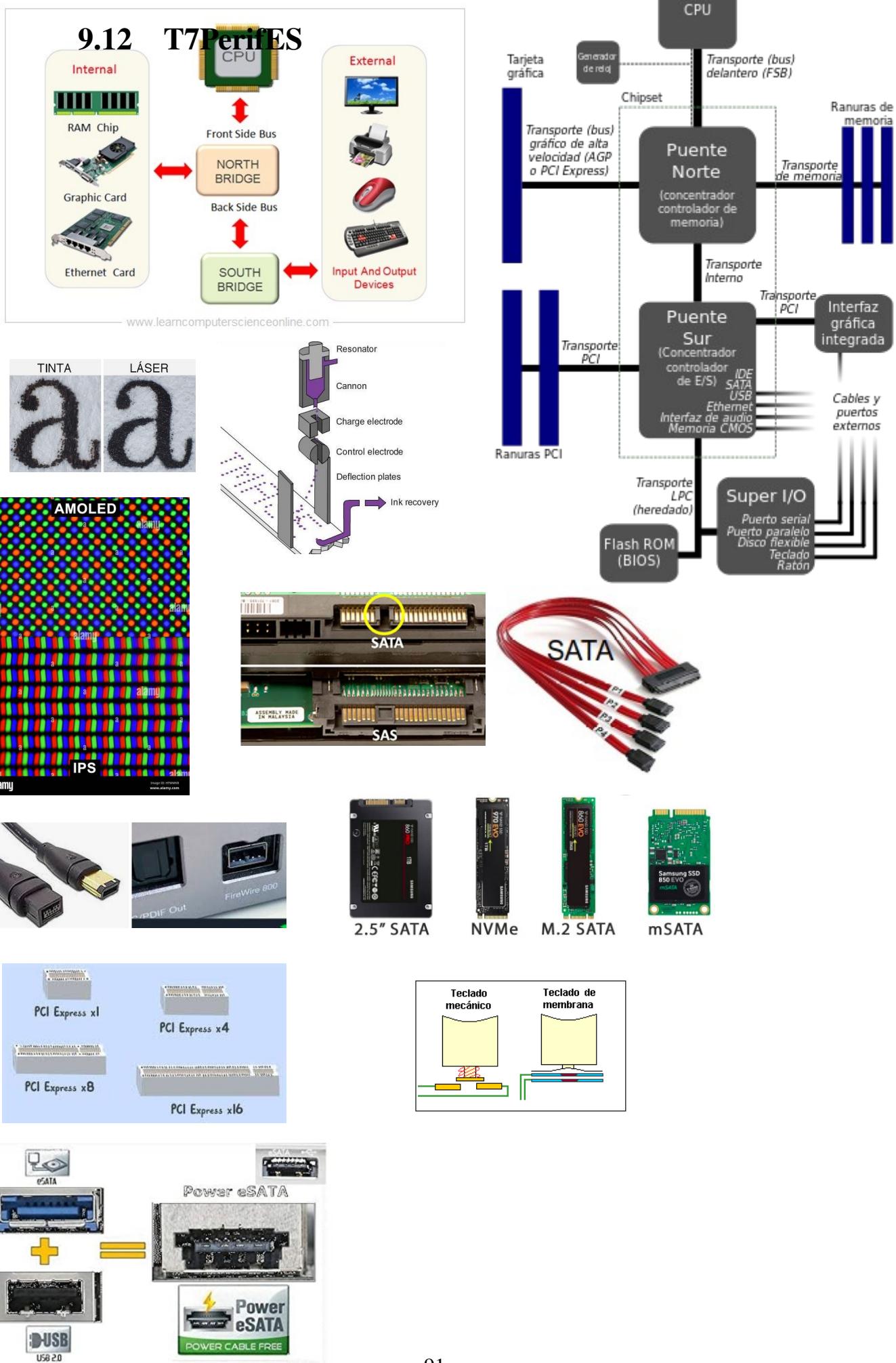
- *Tarj. Audio:* convers. *Analog./Digit. Puls.* PCM con *Filtr???*
 - Micrófono: Transduc. *Sonido* → *Membrana* → *Signal*. Hoy con *Recon.* *Voz* para Dictar o Comandos *IoT* (Futuro Chat-GPT)
 - Altavoz: Calidad *Filtr.*
 - *Jack* : Conect. Otros **TOSLINK** (A. por *Fibra*), Cannon, *Music.* MIDI (parece Cannon). Ver Hijacking (*Cookie*)
- *Tablet* : de bajo *Rend.* llevan *RISC*. Hay varios tipos.
 - Pantalla Táctil: I/O Info según posición 2D del dedo. Hay a) Resistivas. b) Capacitivas. c) Onda acústica por superf.
 - T. *Digit.*: con un lápiz, la mejor es la Wacom Cintiq 22 usada en Disney, Delineantes
 - Lápiz Óptico: es antiguo, un *FotoSens.* (o ElectroSensor de los e^- del barrido de pantalla CRT) y con un cable transmite posic. al PC

Ver *Movil.*

- *Hardw. Red:*
 - *Modem*: Actualm. no Modulan
 - Adapt. Red (*Distors.*)/*Bluetooth/Wifi*: compartir *Recurs.*
 - *WebUSB (IoT)*
- Equipos MultiF.: *Impr.*, Escáner, Fax, ... con *MS* Disco o *USB*. Ej: mi MFC

9.11 Dispositivos de almacenamiento masivo

- *MS*: Almac. *Masiv.*, *Estr.* (Soporte + *Ud.*), *HDD* o *SSD (M.2)* o Petaca. *Disponib.* *RAID (Redund.)*
- IMPORTANTE: Perif. vs Soporte: (*MS*)



Chapter 10

8. Hardware comercial de un ordenador. Placa base. Tarjetas controladoras de dispositivos y de entrada/salida

- *Bus*: B. Sist. UniBus, ...
- *Placa* Base o Madre: PCB que interconect. Compo.. Se comunica con *Bus* a Frec. Reloj *MHz*. Las marcas más famosas son ASUS y Gigabyte, ASROC (barata) (no confundir con Empresa Kingston).
- *Chip set*: conj. de C. de la *Placa* para *Comunic.* *CPU*, *Mem.* y *Perif.* Ordenados por *Vel.* como *Jerarq. Mem.*:
 - Puente Norte (*Bridge*): comunic. CPU con *RAM* y *PCIe* (por Front Side Bus (FSB) antes, hoy con *HyperTransport*). Muy *Vel.*. A veces llamado *MC Hubs* (Solia tener el *iMC*) Fig. 7.12:
 - a) *RAM (MultiCanal)*,
 - b) *PCI Express (GPU, AGP)* Después de *Micro* y *GPU* es el C. mayor de Placa visible. *Actual.* viene ya integrado en el *Micro* y \nexists .
 - Puente Sur: comunic. P. Norte con Dispos. Lentos (no se comunic. direct. con CPU) Fig.:
 - a) *PCI* normal.
 - b) *SATA* y *USB*
 - c) *Super I/O, Serie* (ratón)
 - d) *BIOS*
 - TPM (Trusted Platform Module): *Win*, hace F. *Hash*, *RSA* o *AES*
- *Tarj.* (Card): suele ser una *Placa* que se conecta a P. Madre (*Recurs.* permitiendo la *Modul. Hardw.*). Cuando es común se *Integr.* en P. Madre en *Chip* como *Controlador de I/O, Config.* con *Instal. Driver*.
 - Ej: *NIC*,
 - Ej: *GPU*
 - Ej: T. *Flash*

- Lector T.: mas lento que CPU. De T. perforadas (ver Intro MS) a T. como el DNI electrónico.

Ver *PCI*

- *NIC*
- **GPU** (Graphics Processing Unit): es un *Proc.* especializado en *Paral. Vel.* que viene sobre *Tarj. Graf* y alivia la Carg. a *CPU* en *Videojuegos*. Ej: FFT (Fast Fourier Transf.) y op. FPU para *Videojuego* o *IA*. CUDA (Compute Unified Device Architecture) es una *API* de *Empresa* Nvidia (ver Tab.).
 - *PCI Express x16*: donde conectamos *Tarj. Graf*. Llevan *Mem. Virt.* por *IOMMU*.
 - *VRAM (MP* usada solo por la GPU ver)
 - iGPU: *Integr. GPU in CPU*. La f en *Proc. Intel* significa CPU con GPU deshabilitada ej. i5f (sin GPU) vs i5 (se calienta más). Ver *iMC*.
 - *Google Colab (Nube)*.
 - *HBM: 3D SDRAM* usada por algunas GPUs

Ver *Videojuego, L. Moore, Render. (3D)*.

Los principales *Empresa* son: AMD y NVidea

<i>GPU Popular</i>	<i>VRAM (GB)</i>	<i>Precio</i>
NVidia GTX 1650	4	
NVidia RTX 3060 (2021)	12	
NVidia 4090 (2023)	1000Euros	
NVidia 5000		
AMD Radeom		antes ATI con DirectX (<i>Render.</i>)

- **Zócalo** (*Socket* o *Slot*): *Conect.* hembra *Matr.* de pines a *Placa Madre*. *Actual.* para *Micro* puede llegar a 1300 pines.
Hay L/PGA (Micro sin/con pines). Los hay soldados (videoconsolas-*Videojuego*) y Arq. Abierta (*Libre* de poner otras CPU). Ej: *PCI*. Ver *RAM*
- **Pot.** : *Energ./Tmp* = Fuerza*Vel.=Volt*Int.=Momento*VAng., casi MIPS o FLOPs (*Reloj*). *Micro* unos 150W Ver *Robot*
 - VRM (voltage regulator module, Mod. Regul. Volt): conversor de Pot. que alimenta el *Micro*. Convierte los 5 o 12V de la Fuente al V. requerido por distintas *CPU*, permitiendo CPUs con distinto V en misma *Placa*.
 - Consumo *Energ.*:
 - **Temp.** Termal Output o TDP (Thermal Design Power, no confundir con *Tmp*): max Pot. necesaria para Tª CPU OK. Importante para Portátil (Laptop). Cantidad máxima de calor generada por un Chip (*CPU* o *GPU*) que el sistema de refrigeración está diseñado para disipar bajo cualquier *Carg.* de trabajo. Rel. con *Pot.* max. consumida por refrigeración.

- * Coef. Disip. (refrigeración líquida)s: del aire es $0.02W/m^2K$ (watos por metro² y grados Kelvin) mientras que líquido $0.12W/m^2K$ (6 veces) y agua $0.6W/m^2K$ (25 veces) [NatePCIbai]. Poniendo en todos disipador más ventilador se duplican. El de Ibai lleva líquido y los caros caros agua.
- * Intel i5f (sin GPU) vs i5 (se calienta más)
- * Noel: 3D HBM no calienta

Ver *Instal. Segur. (Tierra, Rack, Disponib.), Secuenc., Cuant., Integr., Encapsul.*

– **Atenuación P.:** ver *Distors.*

Ver *Masiv., PLC Red, Rend., Electr.*

Una bombilla emite $P_0 = 3W$ de luz \Rightarrow a $r = 1.5$ m la intensidad (*Pot/Area*) es $I = \frac{P_0}{4\pi r^2} = 0.1W/m^2$ [Gettyspg841]

Chapter 11

+*9. Lógica de circuitos. Circuitos combinacionales y secuenciales. (17SepJavier)

11.1 Introducción. Conclusión

- Ej. Motivador: ¿como funciona un Mando a Distancia (o máquina expendedora de tickets, *Semaforo*) o *CPU*? Sol. Jerarq. *Chomsky* de *Autom.* ¿Porqué digitalizamos? Sol. Tesis *Shannon*
- Hist. A.: Leibniz, Genialidad *Boole* Nyquist *Muestr.*, Genialidad *Shannon* ventajas de la *Digit.*, *Turing*, Maq. Moore/Mearly, *Chomsky*
- Futuro: *Autom.* Celulares *Wolfram* *Cuant.* (IDEA)

11.2 Circuitos digitales

- *Analog.* vs *Digit. Binar.* [Leibniz] Ventajas *Shannon*. Hoy se mezclan pero ganan D.
- Electr. *Digit.:* permite *TAP* en señales. Distinguimos *Combin.* y *Secuenc.*

11.3 Operadores lógicos

- *Op. Log.:* hay 16 Binarias. Se puede representar de 4 formas: a) *Algebr.* o *Log.:* símbolos $+, *, -$ para luego *Axiom.* *Boole* u *Op. Progr.* $(!, |, ..)$. b) Puerta: *Circ.* (electr.) c) *Tabla Verdad:* dar explicación “AND solo 1 si todo 1=multiplicar”. d) *Diagr.* Venn: *Conj.*. Ver Fig. 11.11
 - **NAND** o Sheffer Stroke $|$: Posee Completitud Funcional (*NOR* también, ver *Axiom.*). Fig. 11.11 con *MOSFETs*. Ej: *CODE2*. Ver *ALU*, *Flash (Integr. 3D)*, *Transistor*,
 - XOR (*Excl.usive Or*): para suma con Acarreo y div. (*CRC*)
 - Equiv. *Wolfram*: se pueden hacer con agua
 - Op. *Conj.* se *Correspond.*

11.4 Álgebra de Boole: axiomas

- *Algebr. Boole* : [The Laws of Thought 1854] (An Investigation of the Laws of Thought on Which are Founded the Mathematical Theories of Log. and Probabilities Hist. A.). Genialidad de Boole: tratar Log. 0 similar a Reescr. Expr. Algebr.. Relacionado con Ventajas de Digit. de Shannon. Es Unif.
- *Axiom. atización*: existen diferentes pero la clásica es: Retículo (PreOrd.) Distribut. Complementario, i.e. EA $(A, +, *, -)$ donde:
 - $+$: supremo, join o límite menor alto (\vee)
 - $*$: infimo, meet o límite mayor bajo (\wedge)

Dados dos elementos $a, b \in A$, como Reticulo tenemos:

- 1) Asoc.: $a + (b + c) = (a + b) + c$ y su Dual $a * (b * c) = (a * b) * c$
- 2) Conmut.: $a + b = b + a$ y su Dual $a * b = b * a$
- 3) Absorc.: $a + (a * b) = a$ y su Dual $a * (a + b) = a$
- 4*) Idempot.: $a + a = a$ y su Dual $a * a = a$ (Proyec.). (Ax. Redund. se sigue de Absorc. pero se suele incluir).

Sea $c \in A$, como Reticulo Distrib Complement. tenemos:

- 5) Distrib.: $a * (b + c) = a * b + a * c$ y su Dual!! $a + (b * c) = (a + b) * (a + c)$
- 6) Complement.: $a + \bar{a} = 1$ (total) y su Dual $a * \bar{a} = 0$ (vacío)

Sin embargo definiendo el Op. *NAND* (Sheffer) basta un solo Ax. según Wolfram $((a|b)|c)|\dots = c$.

- *Leyes Derivadas*:

- L. Morgan: $a * b = \bar{a} + \bar{b}$ (*NAND*) y su Dual $a + b = \bar{a} * \bar{b}$. Se generaliza a n variables
- Tercero Excluido: $\bar{\bar{a}} = a$. Constr. o Intuic. no lo acepta Ej: **Pot. Irr. Rac.**.¹

11.5 Circuito combinacional. Karnaugh

- *Circ. Combin.* o *F. Log. 0*: $f(\vec{x}) = \vec{y}$, i.e. no tiene Mem., la O. está completamente determinada por la I. indep. del t. (Secuenc.). Autom. min. en Chomsky (Fig. 1.7). Ojo a Cond. Carr.. Se puede representar de 4 formas como *Op. Boole* (Puertas, Tablas,..)
 - *Esp. acio F*: Ej: 16 Op. Log. Conj. F. de un conj. X a otro Y . Se llama Esp. porque ej. en *Algebr. Lin.* el conj. de Apl. forma en si un EV. (Serieizar las Matr.).
 - Obj. *Expon.* : en T. Cat. generaliza el Esp. F. El n° de F. de $X \rightarrow Y$ es $|Y|^{|X|}$. Ver *Arbol L. Moore. Curry, Erik Meijer*

Ver Busq. C. (Fuerza Bruta), SFD (Dis. Top-Down), Nivel, Tabla Decis. (Descr.), Prod. Cart., Truco C., Ley

¹“Dem. que $\exists a, b \in \text{Irr. t.q. } c = a^b \in \mathbb{Q}$.”

- Forma *Canon.*: forma Expand. de F. *Combin.*. Hay 2 *Equiv.* por L. Morgan:
 - Suma Minterms: Comb. *Lin.* de *Binar.* ej. $f(a,b,c) = \bar{a} * b * c + a * \bar{b} * \bar{c} = \sum(2,4)$.
 - Prod. Maxterms: ej. $f(a,b,c) = (\bar{a} + \bar{b} + \bar{c}) * (\bar{a} + b + c) * (a + b + c) = \prod(0,3,7)$
- *Karnaugh* : para *Reescr.* más *Eleg.* la *Expr. Algebr. Canon.* y usar menos *Recurso*. Si n var $\Rightarrow 2^n$ casillas \Rightarrow AGrup.aciones de 2, 4, 8 y 16. Si $n > 6$ usar tablas Quine-McCluskey. Si aparecen Indeterminados (*Null Perd. Entropia*) usar a conveniencia.

Ej: col. de 1s, orden adyacentes (cambia solo 1 bit).

$$f = \bar{a}\bar{b}cd + \bar{a}bcd + abcd + \bar{a}\bar{b}cd$$

ab / cd	00	01	11	10
00	0	0	1	0
01	0	0	1	0
11	0	0	1	0
10	0	0	1	0

11.6 Ejemplos de combinacionales

- *Cod. Combin.*: $|I| > |O|$ (I: One Hot; O: n° *Binar.*). Ej: **Conv. Decimal a Binar.** o Teclado o *ROM*. Fig. 11.11
- DeCod.: $|I| < |O|$ (I: Binar; O: otro). Ej: **BCD a Display 7 Segmentos** o Ud. *Control*. Fig. 11.11
- De/*Multiplex*. : dado que *Medio* es caro, distintos *Transm.* o *Usr* comparten el mismo. Diferentes tipos:
 - **M. div. Tmp** (TDM): *Signal Control* selec. quien pasa periódicamente Fig. 11.11. Turnos rápidos. Ej: SONET (*FTTH*), *Telef.*, *Movil*
 - M. div. Frec. (*FDM*): aprovechar A. *Banda*
 - M. *Cod.* (*CDM*): ver *MAC*
 - M. div. Onda (*WDM*): ver *Fibra*
 - Otros: M. Alg.: *MAC*, *Rut.*, M. *Concurr.*: si mismo *Recurso*.

Ver *Hist. Linux, Conmut.*

- *ROM*: Combin. *Cod.+Decodif.*
- Comparador: bit a bit comenzando por el de mayor significación.

11.7 Circuito secuencial

- *Circ. Secuenc.* o *Autom.* Finito: Sist. *Recurs.*, el O. actual dependen del I. actual y del Est. interno. Constan de 2 partes: *C* (*Combin.*) y *M* (*Mem.*) Fig. 11.11. Todo *Autom.* se hace con C. S. Se dividen en:

– Según *Sincr.*:

- * a) *Sincr.*: los cambios se hacen cuando se recibe una señal de *Reloj*. Vent.: más simples que *Asincr.* (OJO en *Transm.* distinto). Ej: *SDRAM*
- * b) *ASincr.*: los cambios se hacen cuando cambia un I. Vent.: más *Vel.* que *Sincr.* y consumen menos *Energ.* al no tener Reloj de GHz. Sol. de *Biolog.. Desv: Cond. Carr.*. Ej: *ADRAM*.

– Según tipo *Autom.* Finitos:

- * a) Maq. Moore: *O Actual.* es F. solo del estado.
- * b) Maq. Mealy: *O actual* es F. de ambos: *Est.* e *I* (menos Est. y reacciona más *Vel.* pero menos segura que Moore). Fig. 11.11)

Ver *Temporalización (Proyec.)*, *Concurr.*, Tma Progr. *Estr., Acces. S. (SAM)*

11.8 Ejemplos de circuitos secuenciales

- Bi*Est.* o Flip-Flop o Latch (no confundir con Estadística *Corr.* o *Prob.*): *Mem.* Bin. básica usada en *Secuenc..* Es no pasiva, necesita Volt, pero poca *Energ.*, usado en *SRAM* y ventaja *Digit..* Aunque hay *JK*, *T*, *D*, el más famoso es el Latch *SR* (Set Reset) de NOR ver Fig. 11.11 y la Tab. Distinguimos:

- *ASincr.*: Latch
- *Flanco Reloj*: Bi*Est.* DDR *RAM*, *Manchester*, *Cod. Land-Pit (Disco CD/DVD)*.
- *Sinc.*: por *Flanco* subida o bajada. b) por nivel alto o bajo

Ver B. *Condic. E. (CODE2)*, *Autom.*, *Secuenc.*, *Cambio Ctx*, *Tabla Pagin.*, *POO*, *Config.*, *Registr. (CPU)*

SR	Q_n	Explic.
00	Q_{n-1}	mantiene el estado que tenía
01	0	reset a 0
10	1	set a 1
11	<i>Indet.</i>	I. a evitar en <i>Cond. Carr.</i>

- *Registr.* de *CPU*: *Jerarq. Mem.* interna, de más bajo nivel (muy *Vel.* y poco *Almac.*) hecha de n Bi*Est.* (*Long. Palabra*) con la que trabaja *ALU* y *Control* dando *ALU Recurs.* (ver). Para *CODE2* tenemos:
 - R. Uso General o *Fich. R.*: hecho con *SRAM*: guarda Dat. o Direc. (*Op.andos*). En *CODE2* almac. ambos tipos, en otras los separa (*Harvard*)??
 - R. Uso Especif.: *AR* (direc.), *PC* (contador), *IR* (intruc.), *SP* (pila) y los *SR* (estado)
- *Registr. Desplaz.*: controlado por *Reloj* y R. SIPO (*Serie I. Paral. O.*, Fig. 11.11).
 - Contador (Circular): módulo 2^n , basado en *Registr. Desplaz.* con *Reloj*. Ver el de 4bits Fig. 11.11. Puede ser des/ascendente. Usados en *Reloj Divisor Frec.* (reducir a rec. lenta=Engranaje) o *Control*.
- **Mando Dist.**: al pulsar Up vamos al canal dependiendo (C) del canal (M) actual. Ver *Serie, TAI, Busq. S. Infrarrojo*

11.9 Autómatas programables

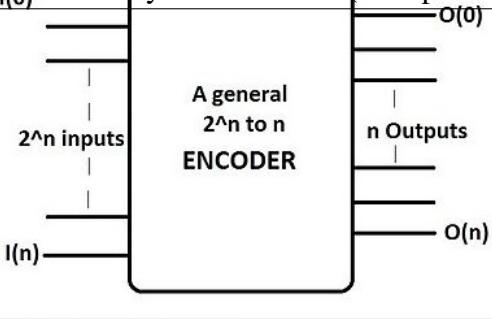
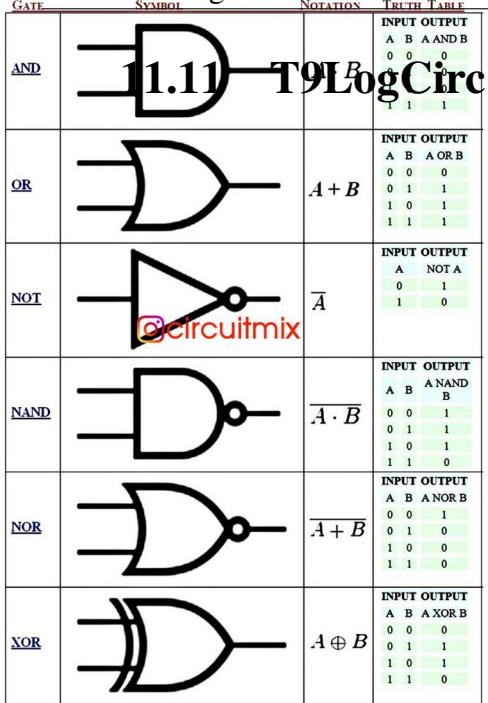
- *Autom.* o PLC (Programmable Logic Control no confundir con *PLC Red*): sist. *Secuenc.* o *Robot, Progr.* con L. *Diagr. Est.* o Tab. Transic. Para Control en *RT* de Sist. *Secuenc.* Industriales. Sus I/O son *Sens./Actuad.* Fig. 11.11
 - Ej: Maq. Expendedora de café (gestión del cambio) según monedas representado con Maq. Moore.
 - J. Chomsky: Los 4 Combin., Secuenc. Pila y Turing
 - Otros A. Cel.: ver *Wolfram*

Ver Ud. *Control, Autocompl., Embeb. RT.*

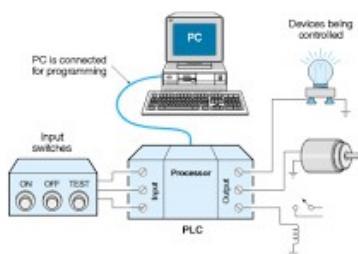
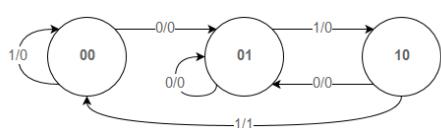
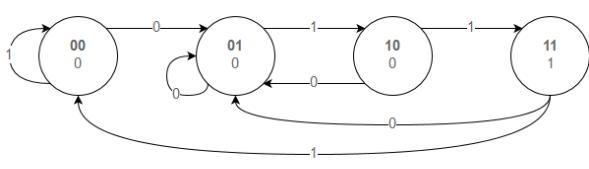
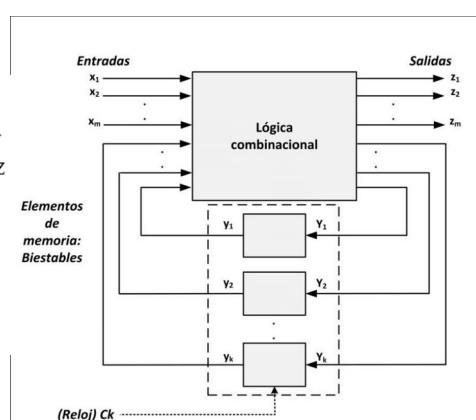
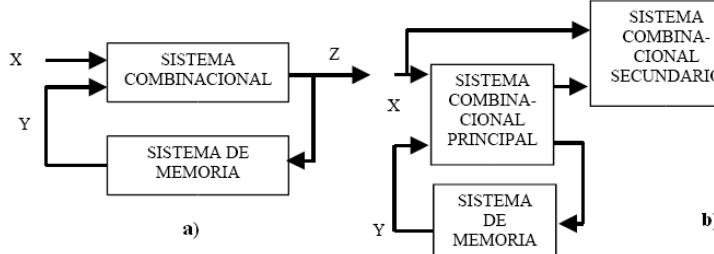
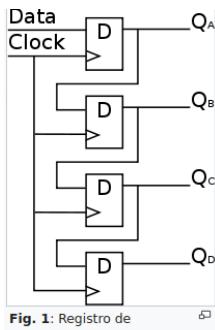
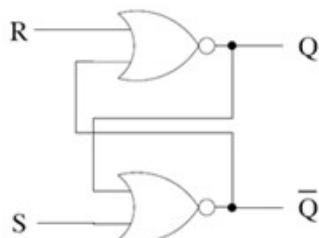
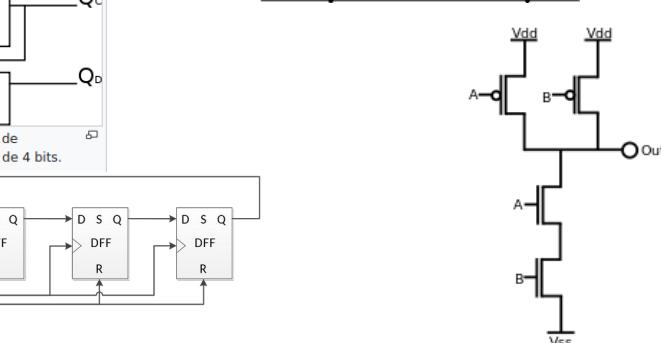
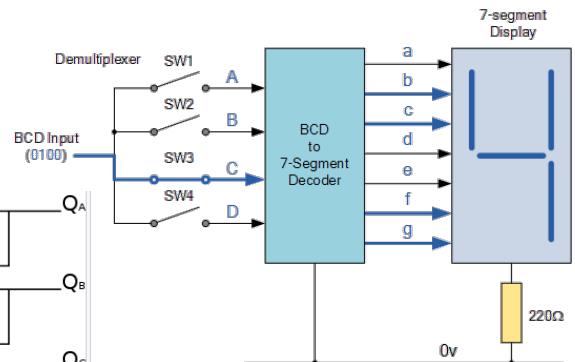
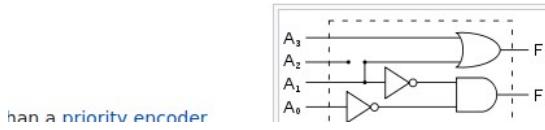
11.10 Recursos, simuladores e IDE

- *IMPORTANTE* completar con Programas de Carrera y Problema Francisco Javier Circ. Secuenc.
- *Simul.* : permiten el *Desarr.*
 - A.: LogicWorks, Orcad, MPLABX, CircuitLogix-PSpice, TinkerCad
 - SO: *Maq. Virt.*, Hyper-Threading (MultiHilo)
 - L.: *IDE*
 - BD:
 - R.: Packet Tracer CISCO

Realidad *Virt.*, Simula (*Smalltalk*), Simulink (*MATLAB*), CASE



A General encoder's block diagram.



Chapter 12

Común SO

- Licencia *Libre* :

- Softw.: Open Source, Access, GNU, Linux, GuadaLinux, IA (Stable Diffusion), Mozilla, Git
- Hardw.: Arq. Abierta basada en *Interop.* (Zocalo CPU). Arduino, RISC-V,
- Softw. L. (**FSF**, Free Software Foundation, Stallman85): siempre se priorizan los aspectos de índole ética (usar softw. como mejor le convenga). Su principio general es el CopyLeft (ver *Ley Europ.*).
 - * GPL del FSF: LGPL del (GNU Lesser General Public License) Licencia para Bibl.
- Cod. Abierto (Open Source Initiative, Perens98 *Hist. SO*): alternativa al FSF. Cod. Abierto se puede vender (*Precio* no gratis) y los usuarios pueden redistribuir mejoras pero bajo la misma licencia. Open Source destaca los aspectos técnicos sobre cualquier discusión moral acerca de las licencias y derechos.
 - * Creative Commons: criticada por Stallman. Es para *Multimedia*
- Ver Ing. Softw. *Inv.*, *Trim (SSD)*, *Basura*, *Free (Dinam.)*, *OpenGL (3D)*, *Jakarta (Progr. Compo.)*
- FreeBDS MIT
- Shareware: Apl. de *Precio* gratis al inicio pero luego pagar.

Ver Asign. Esp. L. (Sist. Arch.), Public., Cod. Pref. L.

- *Virt.* : correr Softw. sobre Softw. y no sobre Hardw. Ej: *Virt. SO*, Hyper-Threading (MultiHilo), *Maq. Virt.*, *Mem. Virt.*, *Red Superp..* Realidad V. *Multimedia* Es un *Unif..* Ver *Vista*, Map. *BDOO* Obj. Rel., *Circ.* V. (TCP), F. V. (Polimorf.). *Simul.*, *Spooling*, *VSAM*, *VFS*
- *Virt. SO* : paradigma donde el *Kernel* permite la existencia de múltiples Espacios de *Usr* aislados llamados: *Container* (LXC, Docker), *Jails* (FreeBDS), Virtual Enviroments, Virtual Priv.ate Server (OpenZV), Partitions. Zones (*Solaris Container*). Es mejora del *Comando chroot*.
 - *chroot PathToRoot command*: lanza el *Comando* en el *Fake Sist. Arch.* (el *Fake Raiz* / cuelga en *PathToRoot* o *Jail*) <https://www.geeksforgeeks.org/chroot-command-in-linux/> Un programa corriendo en SO ordinario puede ver todos los *Recursos* (Ficheros, Perif. conectados, CPU, red, ..), uno en el Container solo los asignados a el.

- *Container*: es una Apl. Autocontenido, i.e. que corre en una *Maq. Virt.* Linux minimal (ej. Debian) sin *Kernel* pero que usa el Kernel de la Maq. Anfitrión (ej. Ubuntu). No es posible Container Win corriendo en Maq. Anf. Linux, pero al revés si porque recientemente Win incorpora Kernel Linux. Podemos tener diferentes Containers en una Maq. Anf. Ver *Almac. Distrib.*, Jakarta EJB (*Progr. Compo.*). Ver *Arq. Kernel*
- *Kubernetes* : es un softw. para **orquestar** (*Admin., Config.* y coordinar sist. *Distrib.* o *Nube, Equil. Carg. Traf.*). Permite coordinar un *Cluster* de Containers (*Virt. SO*) automáticamente para *Despl. Softw.* y *Escal.* en Alta *Disponib.* y en la *Nube*. Relacionado con *Persist..* Su *Servid. API* se comunica con *JSON*. Es *Unif. SO+Red.*
- NFV (Network *F. Virt.*): F. en uno o varios Contaniers (*Virt. SO*), tienen un orquestador. Es futuro de la *Integr.*.

12.1 20. Explotación y Administración de sistemas operativos monousuario y multiusuario.

12.1.1 Administración y configuración de SO

- *Admin.* : *Control*, gestión, *Config. Efic.-Opt.* y *Segur.* de algo.
 - A. *Dispos.*: en Win (Admin. de dispos. con *Comando devmgmt.msc*) (*Driver* o controlador de pantalla), la *GPU* luego lleva su propio controlador (puedes ver su *VRAM*).
 - A. *SO* (o A. *Sist.* o *sysadm*): *Config.* de *SO*, Admin. *Arch.* (*Naveg. MS*), *SO Servid. (Recurso* (impresora), *Permisos Usr*, *Rest...*). *Comandos GNU, DCL*. Importancia según ADDECO (empresa de *Recursos humanos*): es de los perfiles más demandados sobre todo *Segur.* en Red. (FP ASIR). *Comando devmgmt.msc (Driver)*.
 - A. *Red*: ver T71AdminLAN.
 - A. *Remot.*
 - A. *SGBD (DBA)*: *DCL, PhpMyAdmin*

Ver *Req.* (UML, *Diagr.* Casos de uso), *Config.*, *Kubernetes*, A. *Arch. (Dir.)*.

- *Config.* ⊂ *Admin.*: modificar los *Param.* de un *Softw.* y guardarlos en *Fich. Master (Est.* en MS) para que cambie su comportamiento al *Rest. Sesion.* Ej: el *SO Registr. Win.*
 - *Fich. de C.*: *Etc, .bashrc (export PATH="/Dir1:\$PATH")*
 - Escritorio o Desktop: *Linux KDE* (Plasma, Kubuntu, Umbrello (CASE)), *Mate (Mint)*, *Cinamon (Mint)*, *GNome (Ubuntu)*. *Comando gnome-session*.
 - *Registr.*: *Win*
 - *GUI*: puede no acceder a toda la C. y usar *CLI*

Ver *Permisos*, *Kubernetes*, Botón WPS (*WAP* y *Vuln. WPA*)

- *Usr* : el *Admin.* los crea. Ver *SO Servid.. user, group and other*.

- **Grup.** s (Group): SubConj. de pertenencia dada en *Etiq.* (no necesariamente *Jerarq.* o con *Diagr.* de Venn) de *Acces.* a *Recursos* y *Autent..* Es *Unif..* Ver *Usr*, *Permiso* de *Compart.*, *Plan.*, *Asign.*, *Cluster*, *EA*, *Rel.* *Equiv.*, *Karnaugh*, *Her.* de *Interf.*, *IETF*, *Par Trenz.*, *Servic.* *Dir.*, *VLAN*, *Cooper*.
- **Etc** Shadow: *Fich.* de *Config.* de *Usr*. Donde 6 es *Hash SHA-512* para *Autent..* *EtcPasswd* guarda info de *Usr* como Telef. (*LDAP*)
- Multi*Usr Concurr.*: en *SO*, *BD* (*DCL* y *Transac.*)
- U. *SGBD* (ver)

Ver *Tip. Dat. Abstr.*, *Autent.*, *Coher.* (*Git*), *Comando sudo*, *Req.* (UML, *Diagr.* Casos de uso) *Dis.* UI vs UX (*GUI*).

- **Actual.** : *Inform.*

- *Op. BD*
- *SO*: da *Segur..*
- **Legacy** (o *Her.*): *Tecn.* antigua pero que se mantiene por compatibilidad con otros. Ver *Instal.*, *Segment*.

Ver *Popular*, *Recurs.*, *Vista*, *Benchmark*.

- **Comando** s: *GNU coreutils* y estand. *POSIX*

- *grep*: es un *Filtr.* de *Expr.* Regular y *Decl..* Por *Thompson* (*Bell, Hist. SO*).
- *cut*: dato texto *Form.*, *Separ.* *Campos* con *Separ.* *-d* :. Ej: *EtcShadow* (ver IFS *Bash*)
- *tar*: ver *ZIP*
- *chroot*: ver *Virt. SO*
- *lpadmin*: *Impr.*
- C. Red: *Ping*, *ufw*,... (ver *Ifconfig*)
- *cron,at*: *Daemon*
- *tar*: *Backup*
- *apt,dpkg*: *Instal.*
- *gnome-session*: *Config.* *Desktop*
- *top,ps*: *Carg.*
- *Ln*: ver
- *ls -l*: *ls -l* 7 *Fich.* (*d(dir),-(file),s(socket),b(block),...* (*Arch. Dispos.. ls 2> /dev/null Redirec. Err. Arch. Dispos.* a *Virt.. lsof* (*Fich.* abiertos, ver *Descr. Arch.*)).
- *stat*: *INode Atrib.*
- *du/df -h dir*: Esp. ocupado o libre-de-partición-donde-está un *Dir.*
- Win: *devmgmt.msc* (*Dispos.*).
- *sudo*: puedo crear *Pila* de *Acces.* entrando de uno a otro *Usr*
- *chroot*: ver *Virt. SO*
- *passwd*: ver *Autent.*

- Otros: *Trim (SSD)*, *fsck*, *scandisk* (*Integr. Arch.*)
- Otros BD: *MySQL*
- *devmgmt.msc*: *Driver*
- *Op.*: *Pipe* |, Ampersand (&, 2º plano *Back* como *Fork*, ver *Esper.*).
- *POSIX*: *Signal SIGINT Ctrl+c*

Ver *Instr.*, *SQL*, *DCL*, *QBE*, *CLI*, *Pipe*, *Git*

- *Script* : *L.* de *SO*. *Bash* para *Unix* y *.bat* para *Win*. Ver *Cluster*, *Test*, *Lote*
- *Bash* : es un *CLI* y *L. Script* de *Comandos*. Es *Interpr.*. Para *Lotes* o *Concurr.* (job+ampersand)
 - *IFS*: *Separ. Var.* Global para *Comandos* como *cut*
 - *Estr. Control Bloq.* (*if*, *fi*, ver).
 - *POSIX* 0=*Stdin*, 1=*stdout*, 2=*stderr* *Descr. Arch.* (*INode*)
- *Mount* : un *Sist. Arch.*.
 - *Almac. Distrib.*: *Samba NFS* (ver *SSHFS*)

12.2 22. Planificación y explotación de sistemas Informáticos. Configuración. Condiciones de instalación. Medidas de seguridad. Procedimientos de uso.

- *Rest.* Restaurar o Reiniciar o Resetear o ReArrancar:
 - *R. Sesion*: al inicio ocurren muchas cosas de *Config.*, como cargas de del *Script .bashrc*
 - *Reboot To Rest.*: Apl. que congela la *Imag. Disco* al instante *Instal.* gracias a que todo cambio *Admin.* se guarda en *MP*. Ej. DeepFree (*Win*) y DeepLock (*Linux*)
 - *Wake On LAN*: encender *Remot.* el PC (*IoT*).
 - *Hibern.* : es una forma de *Persist.* de la *Imag. Disco*, todo lo de la *RAM* se guarda en un *Fich..*
 - *Imag. Disco* (*Imag. Disco*): *Sist. Arch.* ISO-9660 del *SO*, para Disco (CD/DVD). Forma de *Persist..* Ver *CD-ROM*, *Checksum*, *Reboot To Rest.*, *Instal.*, *Hibern..*

Ver *Sesion (Autent.)*, *SSD*, *BIOS*

- *Instal.* *Softw.*: no confundir con *Despl.* que es en muchos mientras este es en 1 solo PC o lugar. Ej: I. *Driver* a *Tarj..* Ej: I. *SO* con *BIOS* (POST)
 - *Win10* en LENOVO (BIOS con F1 o F12): BootMode 2 tipos UEFI (nuevo): elegirlo para quemar y arrancar el USB con RUFUS (mejor que con Linux Mint). *Legacy* (viejo): probar que lea el USB 1º con UEFI y si no probar este.
 - *Win*: Quemar *Imag. Disco ISO* con RUFUS (no con Linux Mint), ej. con esquema GPT y Sist. Destino UEFI no CSM Instalado en *Legacy*, instalar con UEFI, arranca desde el USB

- *Form. Part.*
- Comandos: *apt,dpkg* para *Linux* y Apl. *InstallShield* para *Win*, descargan *Depend.*, *Vincul. Bibl.* Ej: *sudo dpkg -install Paq.deb*: I. *Paq. Ubuntu*, *sudo apt install Paq.deb*
- *Docum. Instal.*: debemos *Preserv.*
- Otros: I. *Segur.* las 4 (*Temp.-agua-fuego-polvo*, *Electr.-Tierra*, *Autent.* y *Ecolog.* RAEE, también *Disponib.-SAI*, sala del *Centro Dat.*). I. *Hardw.* (*Herram.*).

Ver *Proveedor, Imag. Disco, Proyec..*

- ***BIOS*** (Basic Input/Output System) (UEFI abajo): *Firmw.* que permite: a) el arranque *Rest.* de un SO y b) Ofrece *Servic.* al SO (*Middlew.*). Una *ROM* tiene el Progr.
 - POST (Power-on self-test o autoprueba de arranque): *Subrut.* que por nº de pitidos sacas el error. Antes del *BOOT*.
 - *GNU GRUB* (GRand Unif.ied Bootloader): *Carg.* de Arranque (*Rest.*) de los SO *GNUs-Unix*, pantalla negra (o azul) para elegir entre *Win* o *Linux*. Desarrollado del paquete Grand Unified Bootloader (juego palabras con Grand Unified Theor).
 - UEFI (Unified Extensible *Firmw.*are Interface): *BIOS* moderna.

Ver *Instal., IPX/SPX NetBEUI*

- *Form.atear Part.* : creación de un S.A. en un *Esp.* de *MS* o *Disco* (crear *Swap*, UEFI..). Paso de la *Instal..*
 - *Segment.*: más típico que *Pagin.* en *MS*.
 - *Fragm.*: evitarla

Ver *Subnet, Cuota, Concurr., Fact., DesCompo., Modul.*

Chapter 13

+*10. Representación interna de los datos. (27DicMarcos)

13.1 Introducción

- Ver T0

13.2 Jerarquia datos, información, conocimiento

- *Dat., Inform. Conoc. y Aprend. Autom.*
- *T. Inform.*

13.3 Digitalización: ventajas de Shannon

- **Digit.** : proceso de convertir *Inform. Analog.* en *I. Comput.* i.e. en *Num. Discret.* para TAP. Fig. 3.11. Dicha Inform. suele ser *Fisic. (Muestr., Modem)* pero también puede ser platónica (Matem. [Buzzard] ATP, Ideas, Alg.) pasar a texto o *L. Form.*
 - **Muestr. /Cuantiz.** : los 2 pasos para D. una *Signal Analog..* Si se respetar la F. Nyquist $f_s \geq 2f_{max}$ (ver *Bell*) y $f_{max} < \infty \Rightarrow$ *Digit. Equiv. Analog.* permitiendo trabajar con la *Transf. y Dem.* Tma *Shannon-Hartley*. Para *Voz* se usa Cuant. *Ley Logar. Mu.* (del *Cod.ec G.711* de la *ITU*). Ver *Modulac. Delta. C. Vect.* (*VQ=GAN Aprend. Autom.* usado en *CELP LPC*), *Resol.*, Alta F. M. (*RT*)
 - Ej *Signal*: ver *Medio Transm., Movil??*

Ver *TV TDT, Modem, Preserv. D., Puls., Opt., Certif. o Firm. D.*

- Genialidad de *Shannon* o Ventajas de la *Digit.* o *Binar.*: las 3 señaladas en la Tesis de *Shannon* son:
 1. *Equiv. Circ. Digit. Boole: Expr.* de forma *Digit.* (vs a *Analog.*) los *Circ.* nos permite simplificar *Karnaugh* usando la Genialidad de *Boole*. *Circ. Log.* resuelven todo lo que el *Alg. Boole* resuelve y *Shannon* lo usó simplificar complejos *Circ. Telef.*

2. *Turing*: no conceptualizó el Ord. *Analog.* y necesitamos *Digit.*
3. *Robust.* al *Ruido*: al ruido, *Repet.* y precisión controlada y simple en *Transistor* on/off que si fuera *TerAria*.
4. *Efic. Energ.*: por Off y gracias minituarización por L. *Moore*. Ej: ver *BiEst.*

Otras ventajas:

1. Otros: *Compo.* con otros sencilla. *Modul.* sencilla.
2. Otros: *Popular*: si fuera Ternaria añadir *Cod.*
3. Otros: *Biolog.*: neuronas.
4. *Hist. A.* Termina Ing. *Electr.* y Matem. en 1936 (con curso *Filos.* sobre Alg. *Boole*), trabaja en el Vannervar Bush Analizador Diferencial (un Comput. *Analog.*). Tras pasar el verano del 37 en *Bell* saca su Tesis de *Master* en *MIT* 1938 donde muestra las 3 primeras Ventajas *Digit.* (en *Circ. Combin.* Relés *Telef.* y crea un sumador de 4 bits). *B.* [Garden] (8 Inteligencias) y [Goldstine] (libro De Pascal a Von Neumann) mencionan dicha Tesis como la más relevante del SX.

13.4 Codificación

- *Simbol.* : elemento de un *L.* (*Num./Palabra/dibujo/espacial/marca/signo/alg.*) que *Cod.* (*Wittg.*) un *Dat.* (en cuyo caso se llama *Var.*) o una idea (*Alg., F.*) Ej: S. *Diagr.* Flujo. “La info no se ...”. Ver *Fuente Inform.*
- *Cod.* : *Correspond.* (*Alg.* o *F. Combin.*) que convierte una *rePresent.*ación *Simbol.* de la *Inform.* en otra representación *TAI* (*Num.*). Es un lo que “se dice” Wittg. “La info no se crea ni destruye solo se C. o *Simbol.*”. Rel. con *Compr..*
 - C. *Sincr. AutoReloj* (no confundir con *Cod. Pref.* ni C. *AutoSincr.*): Ej: C. *Manchester*.
 - *Cod. Pref.* (*Libres*): todo C. *AutoSincr.* como el *Unicode UTF – 8* es C. P. y no debe confundirse con C. *Sincr. AutoReloj* como *Manchester*. No requiere palabra para *Separ.* la secuencia de palabras. Ej: *Huffman, Shannon-Fano* y *Omega*. Similar a *Cod. AutoSincr.*).
 - Otros: C. en *Lin.* (*Manchester*), *Cod.ec* (*Multimedia*), *Modulac.*: FM,..., Land-Pit (*Disco CD Flanco*)
 - Otros: 4 *Tip.* clásicos de Cod. son: *Num., Texto, Instr.* y *Multimedia*. Mediante la *Equiv. Anal. Digit.*, al final podemos reducirlo una *Tupla* (Conj. Ord. Num.)

Ver *Sincr., Huffman, Traduc., Ud. Control, Cript., Music.* MIDI, C. César (Cript. *Simetr.*)

13.5 Representacion de números

- *Cod. Num.* : de un *Simbol.* permite *TAP. Creer en Ficciones* [Harari]: un *C* o un *vect* es un *Alg.* sin Input. Ver *Digit., Ley*

- Sist. Num.: reglas + *Simbol.* (*Digit.*) para rePresentar cantidades. Distinguimos: a) SN No Posicional: ej. Egipcio (4+tacho). Romano es mixto?? $IX \neq XI$. b) SN Posicional: se basa en un conj. finito de *Simbol.* o Alfabeto y un valor relativo o pesado según posición, conocido como **Tma Fundamental Num.** (TFN) (Base Expon. y Comb. Lin.) $n = \sum_i d_i * b^i : d_i \in \{0, 1, \dots, b - 1\}$. Ej. babilónico y los siguientes
 - *Binar.*
 - Octal: truco de Bin. a Octal. tomar de 3 en 3
 - Decimal: el de siempre
 - Hexadec.: $\{0, 1, \dots, 9, A, B, \dots, F\}$ para *Direc. Mem.* como *Punt.*
- **Binar.** (2 Aria): $\{0, 1\}$. Un Bit es cada posición [Leibniz] *Hist. A..* Ej: “Contar 1024 con los dedos”. Ventajas: ver *Digit. Shannon*.
 - Metr. o Ud. *Inform.*: el **Byte** para *Direc. Mem.* y el **bit** para *Transm..* 1 **Kibibit (KibiByte)**= $Kib = 1024 = 2^{10}$ bits (Bytes) según ISO/IEC 80000-13, para distinguir del Sist. Internacional donde 1 **Kilobit**= $Kb = 1000$ bits, aunque muchas veces se aproximen. Otras $Mib = 2^{20} \approx 10^6$, $Gib = 2^{30} \approx 10^9$, Tib, Eib, Zib, Yib .

Ver *Arbol B., QBits (Cuant.), Ensambl., Fich. B.* (ver *Texto*). *MS, MP*

- Convers. entre Bases: De $b_A \rightarrow b_{10} \rightarrow b_B$: pasar siempre por base 10-. De $b_x \rightarrow b_{10}$: pesar por base. De $b_{10} \rightarrow b_x$: a) dividir y alrevés o b) encajar potencias de base.

Ej: $17 = 16 + 1 = 10001$

- RePresent. Num. Enteros (dados N bits y nº n):

- **Bin. Puro:** solo permite $\mathbb{N} \supset \{0, \dots, 2^{N-1}\}$.
- **Signo-Magnitud:** $S(1b) + magnitud(N-1b)$. solo $\mathbb{Z} \supset \{-2^{N-2} \dots -0, 0, \dots, 2^{N-2}\}$. Desv: 0 doble represent.
- **Compl. 1 (C1):** Positivos como Signo-Magnitud 0[*magnitud*] (Bit Izdo a 0 y resto magnitud). Neg. *NOT(positivo)* (neg. es el complemento del pos.). Ventaja: sumas y restas con circuitos de suma. D: 0 doble representado.

$N = C1$	$N = C2 (C1+1)$
$0 = 000$	$0 = 000$
$1 = 001$	$1 = 001$
$2 = 010$	$2 = 010$
$3 = 011$	$3 = 011$
$-3 = 100$	$-3 = 101$
$-2 = 101$	$-2 = 110$
$-1 = 110$	$-1 = 111$
$-0 = 111$	$-4 = 100$

- **Compl. 2 (C2):** $C2 = \begin{cases} C1 & si N > 0 \\ C1 + 1 & si N < 0 \end{cases}$ (acarrear). La *Impl.* en PC *Actual*.
- **Exceso o Sesgada:** $n_s = n + S$, S t.q. \nexists nº negativos. Ver *BCD-Exceso3*

- **BCD** (*Binar. Cod. Decimal*): cada digit. [0, 1, ..9] en *Binar.* con 4 bits (se extiende a *Texto* ver). Ej. 943 = 1001, 0100, 0011. V: *Cod.* rápida sin perder precisión, útil en *E/S.* D: op. circuito complejo. Tiene variantes donde los dígitos no se representan en *BCD* natural (pesos 8, 4, 2, 1) si no con otros pesos. Aiken (2, 4, 2, 1 para restas y divis.) Exceso3 (*Sumar3* a *BCD* natural). Ver Fig. 1.6. Ver Cod. B. Combin.. Ver Tarj. Perf. (*Texto*)
- Represent. *Num. Reales*:
 - **Coma Fija:** $n = \text{entera}.\text{decimal}$ 1^a computadoras. D: por tamaño *Palabra* fijo, no permite n° grandes y pequeños
 - **Coma Flotante:** $n = M * B^E$ con *M* mantisa, *B* base y *E Expon.* V: aunque aumente max y min, densidad precisión logarítmica. Norma [IEEE 754](#) : con *Palabras* de 32 (simple prec.) y 64 (doble) bits (1b de signo, $8/32 = 25\%$ y $11/64 = 17\%$ exponente, 23 y 53 parte fraccionaria de mantisa) y media o cuadruple precisión.. Def. *NaN*.
 - *Hist. A.*: FPU la incluyeron ya *Torres* y *Zuse*. (ver)
 - ***NaN***: definido por (*IEEE 754*, no confundir con ∞). Ver *Null, Manej., MATLAB*

Ej:

Recordar, poner notación científica binar. Dividir (por 2^n es mover coma). Prod (sumar expon). Suma (poner base y expon común)

13.6 Representación de texto

- **Texto** : conj. *Palabras. Cod.* de caracteres: alfanuméricos y especiales. Con *n* bits 2^n *Simbol.. Estand.* en:
 - **BCD-IC** ($6b = 64$): caract. *BCD Interchange Code*, no caract. especiales, adapat. a $6b$ del cod. *IBM* de *Tarj. perforadas* (punched cards) *Hist. A.. E-BCD-IC* ($8b = 256$ caract. *Exten. BCD-ID*, minusculas, may y especiales).
 - **ASCII** (aeski:): $7(+1b - \text{paridad}) = 128$ caracteres (*Err.*). *Form.* estándar del T. Plano. La *Exten.* de $8b$ es compatible con *Unicode UTF – 8* (ver). Por *ANSI*. Ver *Separ. EOF, \n*
 - **Unicode** : $16b = 65536$ caracteres (latino, griego, árabe, chino, etc). Hay varias *Vers. UTF – 8/16/32*, pero la *UTF – 8* domina el 98% *Web* y es *Cod. AutoSincr. (Cod. Pref.)*. Creada por *Thompson* (de *Unix* en *Bell Hist. A.*) debe su éxito a que es compatible con *ASCII* de $8b$ y fué adoptada por el *IETF. Estand.* por muchas *Empresas* como *IBM, AppleMac, MSWin, Oracle, HP,..* También en muchos *L.* como *JavaScript (Naveg.), XML, Java, LDAP, CORBA (Obj. Distrib. Progr. Compo.)*. Es la aplicación oficial del *ISO/IEC 10646*
 - Otros: *ZIP* y *RAR*, *Web-HTML=Imag. Vect.*
 - T. Plano: usado *ASCII* y *Unicode UTF – 8*. Ej: *RFC, Fuente, Cript..*
 - **PDF** : es un *Container*
 - *Caden. o cadena Texto*:
 - *Fich. T*: contrasta con *Binar.*, ver *Separ. (Registr. Fich.), C*

Ver *Req., Docum., Expr. Regular*

13.7 Representación de estructuras de datos

- *Estr. Dat.*: *Array*, *Texto*, *Lista*, *Arbol*, *Graf.*, *Fich.*

13.8 Representación multimedia

- *Multimedia*: *Fich.+Cabecera*, *Cod.ec*, *Container*
- *Signal Audio* : $Ampl(t) = \text{vector} = \text{array} - 1D$
 - Tamaño de Inform.: Nº canales (estereo/mono), Freq. *Muestr.* ($> 2f_{max}$), *Cuantiz.* de ampl. (bits/muestra), duración
 - Ej: *Compr. MP3* (efecto enmascaram.), *MIDI*. Sin compr. *WAV*. *AAC* (*Cod.ec* del *MP4*)
 - Ej: *VoiceXML* conj. palabras para hacer *Interf. Voz*, *Gener. Proced.* (*Videojuego*).
 - *Voz* : sinónimo de A. por mismo A. *Banda*. Ver *VoiceXML*, *Recon.*, *Autent.*, *Su-**plant.*, *Kinect* *Videojuego*, *Dis.* Top-Down, *VoIP Triple Play*, *Telef.*, *Movil*, *MP3*, *LPC*, *Bluetooth*. *Cuantiz.* *Ley Mu.* *DSP*, *RT*
 - *MIDI* (*Music*. *Instrument Digit. Interf.ace*): es un *Estand.* Japones como Notación Musical(tono)+Intens.+vibrato+.... Una canción se *Cod.* en unos cientos de líneas, algunos KB. También *Estand.* *Conec.. L. M.* es *Form..* Ver *Videojuego*, Converg. *Tecn.*

Ver *Socket*, *Conec.* TOSLINK Jack o (*Fibra*), *Bibl.* PortAudio ver *ALSA*. *RT Tele*

Una señal de 10 seg. con $F_s = 16kHz$ y *Muestr.*as de 2Bytes ¿cuanto ocupa?

- *Imag.* : $\text{Color}(i, j) = \text{matriz} = \text{array} - 2D$ *Array+PaletaColores*
 - Tamaño: *Resol.* (pixelado, *Muestr.*) y *Cuantiz.* del color (bits/pixel o profundidad)
 - Paleta de Colores: B&W, RGB, o CYMK (Cyan, Yellow, Magenta and blacK)
 - Mapa Bits o Bitmap (el de antes): Ej: BMP, PNG (BMP *Compr.* sin perdida, mejora GIF), JPG (compresión con perdidas), TIFF, GIF??
 - *Map. Graf. Vect.* ores ($\sum_i f_i(x, y)$) donde las f_i son elementos *Geom.* básicos (círculos con centro y radio, lineas, polígonos, texto...) y *Fract. Recurs..* Ocupan menos espacio y al zoom (o escalado) no pierde resolución, pero menos realistas (sombra...). Bueno para imágenes simples: Logos, *Diagr.*, **Fractal** P5.JS JavaScript... Ej: EPS, SVG. Ver VQ (*Aprend. Autom.*, CELP *LPC*), *Array*, *One Hot V. Instr.* *Exten..* Es *Unif.* por *Texto Web HTML*. El mundo *Fisic.* puede ser así!! (ver)
 - Paletas: *RGB* (color aditivo) o *CYM(K)* (color subtract.) (Cyan, Yellow and Magenta (and Black)). Fig 13.11

Ver I. *Disco* (ver *ISO*).

- *Model. 3D* (o *Graf.icos*): *Alg.* que *rePresent.a* un *Obj. Geom.* de 3 *Dim.* y que se *Render.* en pantalla. Archivos tipo *.stl* o *.obj* leidos por *JavaScript P5JS*, *SolidWork*, *Blender*.

- *Impr.:* calibrarla, extrusor, y mueve base. Los pasos son Tinkercad → Cure (poner pata tmp para que no se caiga brazo fino, rellenos, huecos..)→ Impr. 3D

Ver Videojuego, Transistor, Hologr. Persist., Integr. 3D (RAM HBM y Flash)

- *Video*: $\text{Color}(i, j, t) = \text{array} - 3D$. Frames de *Imag.* dan ilusión de movim.

- *Cod.ec AVC del MP4*
 - Ej: AVI, MP4, WMV,FLV, MOV

Ver *Videojuego*, V. Bajo Demanda (TV), V. RAM (I/OMMU).

- Realidad Virt.: ..

13.9 Compresión

- **Compr.** : Compresión Equiv. a Comprensión según TAI. Busca similitudes Esp./Tmp (Markov/Recurs.) . C. Populares:

- **RLE** (Run Length Encoding): C. Dic. para Texto Ej: $aaaaaaaa \rightarrow a * 10$ ocupa menos. Ej: $BBBBBBBBBBBBNBBBBBBBBBBNNNNBBBBBBBBBBBBBBBBBBBBNBBBBB1N12B1N12B3N24B1N14B$

- *Huffman*: las que tienen alta Prob. de aparecer, poco *Cod.* $a \rightarrow 0$ no 0000

- **ZIP**: según un num. usa los *Alg. C. Dic.*: 1 (LZW, Lempel Ziv Welch), 2 – 8 (LZ78 o LZ77 + *Huffman*), etc. También es un *Container*

El *Comando TAR* toma muchos *Fich.* y crea un tarbar (Fig. 13.11) luego permite comprimirlos con el programa GZIP (*GNU ZIP*) dando un *.tar.gz*. GZIP lleva *CRC*. Otros que C. a ZIP: PK-ZIP (compresor para *Win MS-DOS e IBM*), RAR, ARJ.

- **LPC** (*Lin. Prediction Code*): form. *Recurs.* y *Markov* $\vec{a}, \vec{b}, \sigma_0$ para Voz. Enviar MT (*Turing*) + *Ruido* pequeño restante cuantizado es mas corto que toda la señal.

- CELP: mejora *LPC* con *Dic.* y *Cuantiz.* *Vect.* *Comb.* *Lin.* de *Palabras* según estad. Va en *GSM Movil*

- **MP3** (MPEG-1 Audio Layer III o MPEG-2 Audio Layer III): C. Pérdidas para *Audio*. Basado en *Transf.* Fourier-Mel-Cepstrum+*PCA* elimina *Redund.* inaudibles por Efecto Enmascaram. Ver *TV TDT, Voz*.

- **JPG** (del JPEG): C. Pérdidas para *Imag.*. Basado en *Transf.* DCT+PCA quedarse con la esquina 4x4 de los 8x8 de DCT

- *Imag. Vect.*: con fractales y *Recurs.*

- **MP4** (MPEG-4 Part 14 =ISO/IEC 14496-14:2003): es un *Container* de los siguientes *Form.* o *codes*. a) *Video* AVC (Advanced Video Coding)=H.264=MPEG-4 Part 10: aplica C. Interframe o Diferencial basada en eliminar *Redund.* entre frames parecidos $f_i = f_{i-1} + \text{cambio}$. b) *Audio* AAC (Advanced Audio Coding), también MP3 c) Subtítulos 3GPP (MPEG-4 part 17)

- **MPEG** Moving Picture Experts Group de la ISO), ej. *MP3* y *MP4*, **MPEG2** (TV Digit.) **JPEG** Joint Photographic Experts Group ISO e ITU-T, ej. *JPG*

Ver *Eleg.*, *Ensambl.*, *Python*, *Cod.*, *Coment.*

Multimedia	Sin C.	Con. C.
Texto	.txt .doc .odt	.ZIP .tar .rar
Audio		MP3
Imag.		JPG
Video		MP4

- *Compr.* Sin Perdidas (Lossless): para *Texto*. Tiene límite de *Shannon ZIP* → ZIP puede crecer por *Cabecera* a) C. *Est.* o *Estoc.*: *Huffman*, *Shannon-Fano*, *aritm.* y **Compr. Predict.** *LPC* b) C. *Dic.* (o *Subst.*): RLE (ver) y LZW (ver) de ZIP (otro: CELP *LPC*) c) C. *Hibridos*: con las 2 fases anteriores (*Est.* + *Dic.*). Los *Actual..*.
- *Compr.* Con Perdidas: para *Multimedia* donde exactitud no importa pero si la idea a) **C. Diferencial** o *Predict.* o *Recurs.*: *LPC*, *MP4 Modul.* *Delta*. Si *Markov Actual.* depende de cercanos b) C. *Transf.* (más *PCA*): Fourier *MP3*, *DCT* *JPG* c) Cuant. Vect. (*Dic.*): *CELP* *LPC* d) **C. Fract.**: trasnf. geométricas afines de Match para *Busq.* *Bloq.* parecidos. Puede usar *Dic..* Ej: *Imag.* Vect. Manderbrot *Recurs.* e) C. en Movimiento (Interframe): eliminar *MP4* f) *Aprend.* *Autom.*: GAN=Autoencoder=VQ
- Otras: Aumenta *Vel.* de *Transm..*

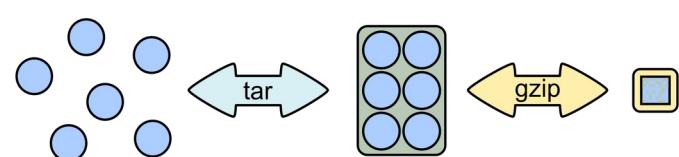
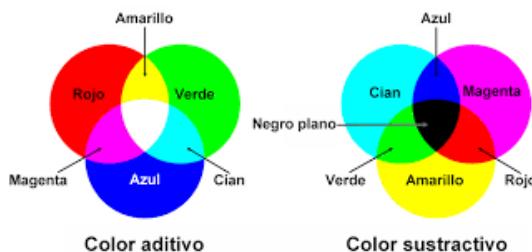
13.10 Manejo de errores

- *Manej.* de *Err.*: *Checksum*, Paridad, *Hamming*, *CRC*,

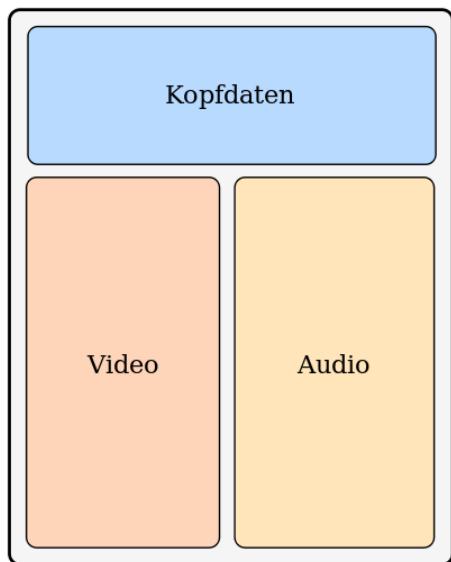
CODIFICACION BINARIA ASCII

13.11 T10 ReprintDat

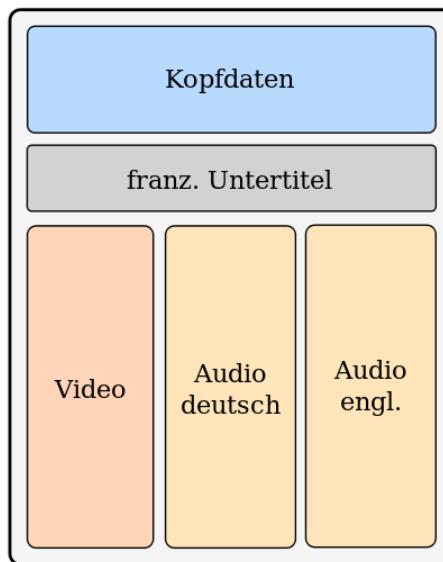
Character	Binary Code						
A	01000001	Q	01010001	g	01100111	w	01110111
B	01000010	R	01010010	h	01101000	x	01111000
C	01000011	S	01010011	i	01101001	y	01111001
D	01000100	T	01010100	j	01101010	z	01111010
E	01000101	U	01010101	k	01101011	!	00100001
F	01000110	V	01010110	l	01101100	"	00100010
G	01000111	W	01010111	m	01101101	#	00100011
H	01001000	X	01011000	n	01101110	\$	00100100
I	01001001	Y	01011001	o	01101111	%	00100101
J	01001010	Z	01011010	p	01110000	&	00100110
K	01001011	a	01100001	q	01110001	'	00100111
L	01001100	b	01100010	r	01110010	(00101000
M	01001101	c	01100011	s	01110011)	00101001
N	01001110	d	01100100	t	01110100	*	00101010
O	01001111	e	01100101	u	01110101	+	00101011
P	01010000	f	01100110	v	01110110	,	00101100



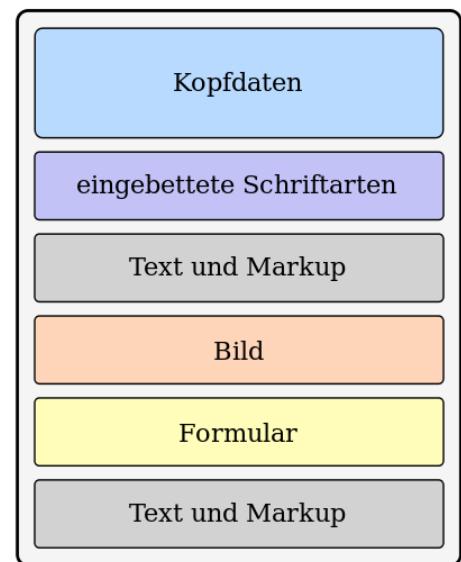
z. B. AVI-Datei



z. B. Datei mit Matroska-Container



z. B. PDF-Datei



Chapter 14

*12. Organización lógica de los datos. Estructuras dinámicas.

14.1 Introducción. Conclusión

- Ej motivador: ¿como crear listas *Dinam.* por *Insert.*? Sol. con *Punt.*
- Ej: ¿como recorrer todos los elementos de un Arbol de forma Efic. para Buscar *Masiv.* en un *Sist. Arch.*? Sol. Arbol B+
- Ej: ¿Busq. Vel. en Sist. Arch.? Sol Arbol B+, ¿para que sirve un Arbol? Sol. *Huffman*, *SPN*, familia, Sist. Arch....
- *Filos. MATLAB*): siempre que se pueda intentar *Lin.*
- *Hist. L. Hist. SO*: Donald [Knuth68] “The Art of Computer Programming” sobre *Anal.* de Alg. de programación y *Arbol. IBM73 Arbol B+ para VSAM*
- Futuro: *NoSQL* como *JSON*.

14.2 Organización lógica de los datos: Tipo de dato abstracto vs estructura de datos

- *Dat.* vs *Var.*

	Commun	Difer.	Otras	Ej:
<i>Tip. Dat. Abstr.</i> (TDA)	<i>EA</i> (conj. dat., rel. y op.)	Def. <i>Form.</i> Log. de un <i>Tip. Dat.</i> / Pto de vista del <i>Usr</i>		<i>Cl.</i> , N° Complejo (def. como EA)
<i>Estr. Dat.:</i>	idem	<i>Impl.</i> F. Fis. de un T. D. / P. V. <i>Impl.</i>	Importa la <i>Efic.</i> de <i>Acces.</i>	<i>Array, Lista</i>

- **Tip. Dat. Abstr.** : define la Forma Log. de un *Tip. Dat.* (comportamiento que ve el *Usr*) de un *Tip. Dat.*. Ej: *Cl.*, *Graf.*. Las *Op.* más famosas son:
 - *Compo.*: de TDA (ver *Estr. Dat.*).
 - *Compare(s,t)*, *Hash(s)*, *Print(s)*. (se hace con *Polimorf.??*)
 - También: *Create(s)*, *Copy(s,t)*, *Delete(s)*.

Contrasta con la *Estr. Dat.* (ver Tab.) que es la forma física (*Anal.* que ve el programador o *Impl.*) del Dat. Ambos son EA (*Conj.* o rango + *Op.* permitidas), pero dif. segú *Filos. Cat.*

- **Estr. Dat.** : EA (con su *Conj.* y *Op.*) que ve el *Impl.*ementador, pensado para *Acces./Busq.=Retrieve=Recuperar (Insert./Del.) Efic.*.. i.e. Rel. o *Estr.* entre *Dat.*, como *Op. BD* y *Form.* de *Almac..*
 - *Compo.*: de E.D. ej. podemos crear un *Vect.* de *Obj.* o *Tip. Dat. Abstr.*
 - Ej: *Array (Texto o Caden.)*, *Lista*, *Arbol*, *Graf.*, *Hash*
 - Ej *NoSQL*: *JSON*, *XML*.
 - Otros Ej $??$: *PCB*, *INode*, miembros *Cl.*, *Mem.* *Asoc.*, 4 *Tablas Básicas SO*

Contrasta con *Tip. Dat. Abstr.* en que la E. D. sirve como base de esta e *Impl.* la *Form.* física de un *Tip. Dat.* (ver Tab.). Aparece en *Asign.* *Mem.* Es *Unif. Equiv.* a *Form..* Ver *BD*, *Fich.*, *PCB*, *INode*, *Struct*, *Serielizear*, *ISAM*

14.3 Clasificación de las estructuras de datos: dinámicas no/lineales

- *Jerarq.* de *Estr. Dat.*: Fig. 14.11 y 31.10:
 - *Tip. Dat. Atom.* o Simple o Primit. o Elem.: Una *Var.* representa 1 solo elem.. Ej: $\mathbb{B}, \mathbb{Z}, \mathbb{R}$ o *Char*
 - T. D. *Compo.* o *Estr.:* de Simples. Una *Var.* representa varios elem. Estos pueden **Index.** (*Enumer.* de *Acces.*) *Efic.* como *Array* o *Struct* o lento como *Graf..*
 - E. D. *Est.atica*: su tamaño ocupado en *Mem.* se hace en T. *Compil.* y no cambia en T. *Ejec..* Ej: *Array*, *Struct* o *Registr.*
 - E. D. *Dinam.:* cambia en Ejec. el tamaño y gracias a *Punt.* o *Lista Enl.* Fig. 14.11
 - * *Lin. Lista*, *Tabla Tupla BD (ISAM)*, *Dic.*, *Pila*, *Cola*, *Fich.??*
 - * No Lin.: *Arbol*, *Graf.*
 - E. D. Homogenea: $\forall \text{element} \in \text{Unic. Tip.}$. Ej: *Array*, *Cadena Texto*
 - E. D. Heterog.: *Struct* o *Registr.*
 - E. D. Interna: almac. en *MP.* Ej: *Struct*
 - E. D. Externa: en *MS.* Ej: *Fich.* y *BD*

La elección depende del problema (cada uno tiene sus Ventajas e I)

14.4 Listas

- *Lista* : E. D. *Lin. Dinam.* (*Insert./Delete*) Homog. Elem. del mismo *Tip.* ¡En *Python* se llaman *Tupla* si mismo T. y si no L.!¹. **Longitud**=nº elementos. Cada Elem. *aPunt.* al siguiente: \forall elem. \exists_1 sucesor/predecesor excepto 1º (*Nodo* cabecera) y último. Ej: *Pila* o *Cola*. L. *Enl.* es *Unif.* (ver abajo). Ver *LISP*, *Enumer. Struct*, *QuickSort* (*Ord.*)
- L. Contigua: elem. en pos. contiguas de mem. I/D \Rightarrow desplazar.
- L. *Enl.*: cada elem. o *Nodo*. almac. 2 *Campos*:
 - a) C. Inform.: con *Dat.* b) C. *Punt.*: con direc. al siguiente Nod. \Rightarrow I/D solo reemplazar (no desplazar todo como en Contigua).
 - El 1er P. señala inicio de L. y último P. es *Null* (Fig. 14.11).
 - En C se hace con *Struct* (ver abajo).
 - Uso: *Tabla* y *Tabla Hash*, *Pagin.* o *Arbol B++*, *INode*.

Permite el *Dinam.* de *Estr. Dat.* en *MS* como *Blockchain* luego la L. E. es *Unif.*. Ver *Asign. Sist. Arch.* (de *Libre* o no), *FAT*.

```
struct nodo {
    int dato;
    struct nodo * siguiente;
}

import java.util.*;

public class LinkedList1{
    public static void main(String args[]){
        LinkedList<String> cars = new LinkedList<String>();
        cars.add("Volvo");
        cars.add("BMW");
        cars.add("Ford");
        cars.add("Mazda");
        System.out.println(cars);
    }
}
```

- *Op.* permitidas (como *Op. BD*): a) *Busq.*: la base del resto de Op. Es B. *Lin.* (*BigO(n)* *Efic.* *MATLAB*) b) *Insert.*: Busq. posic. y reescribir Punt. (el Nod. anterior al actual y este al siguiente). c) *Delete*: d) *Map.* o Recorrido: misma Op. a todos elem.
- L. Doblem. *Enl.*: se apunta además al anterior para recorrerla además hacia atrás, más *Efic.*. Fig. 14.11.
- L. D. E. Circular: último elem. apunta al 1º (*Transf. Fourier*), Fig. 31.10
- L. de Listas o MultiL.: cada Nod. *aPunt.* a otras L. *Recurs..* Ej: *Graf.* o *Array* pero diferente tamaño $[[2,4,6,8,10],[3,15],[4,8,16,20]]$ Ej: Celdas en *MATLAB* o *Arbol* o *Graf..* Fig. 31.10.

¹Una L. es una *Tupla* modif. pero menos *Efic.* en Mem. que *Tupla*. Según *Filos.* [Lamport] solo hay *Tuplas* y *Conj..*

14.5 Listas especiales: pila, cola y diccionario

- **Pila** : *Lista LIFO* (Last Input First Output), i.e., Las Op. I (PUSH o Apilar) /D (POP o Desapilar) se hace siempre por la cima o tope. Otras Op. son Read (del tope) y Crear/Destruir P. Ej: *Python L.append='hola'*, Ej: *Comando sudo* Se usa para *Recurs..* Ej: *Punt. P. (SP CODE2)*.
- **Cola** (Queue, Teoría C. ver *Traf.*): *Lista FIFO* de *Petic.* mientras que el *Buffer* es más de *Dat..* Es necesario mantener un *Punt.* extra de frente y otro de final (Fig. 14.11) Op. *Efic.* de: a) Read (**del final!**). b) I (encolar elem. **al final** de C.) / D (desencolar **1er** elemento). Otras: Crear/destruir. Las hay con *Prior.* Fig. 31.10. Es *Unif..* Ver *Congest., Comut. Circ., Prior., N-SCAN (Brazo)*.
- **Dic.** (Diccionario o Array Asoc.): es una *Estr. Dat.* o tabla con Claves (*Key, Etiq.*) y *Dat..* Desde *Python 3.6* los D. deben tener sus Claves (*Key*) ordenadas. *Struct* es para colecciones o *Array* de E. D. mientras que D. es para largas *Lista* de elementos nombrados. Las 2 formas de implementar *Busq. Efic.* es con *Hash* o *Arbol B. B. AutoB.* Ver *Dic. BD, Compr., CELP o LZW (Lempel Ziv ZIP)*

14.6 Árbol binario

- **Arbol** (Tree): *Estr. Dat. No Lin.* que representa una *Jerarq.* padres e hijos. Cada *Nod.* a *Punt.* a varios.
 - Def.: *Graf.* conexo donde: a) $\forall no_i \exists_1$ padre (y $\exists > 0$ hijos). b) con min. n° de Arcos (cada a_i es un itsmo, deja de ser conexo si lo eliminas). c) sin *Circ.* o Bucles (ver A. *Recubr. (STP)*)..
 - Creac. Def. *Impl.:* a) *Obj. Recurs.:* *Dis.* Bottom-Up empezando por hojas (ver *SPN*). b) *Matr. (Array):* como *Graf.* Nod. y Arist. pero no restringe *Estr.* c) *Struct en C* (ver A. *Binar.*)
 - Def.: *Raiz/Hoja:* *Nodo* de *Nivel* extremos. *Gr(no_i)*: n° hijos. *Camino* (longitud): *Tupla* (n°) aristas. *Ambito:* si nodos conexos en diGraf. *Profund.:* n° de *Nivel* o Height (ver A. B. B. auto B.). *NNodos = n:* tamaño es *Expon.* con *Profund.. Peso:* n° hojas (ver *Rot.*).
 - Op.: *Rot., Busq., Insert./Delete., Crear*

Ver *Bosque, Dir., Topol. Red, Concurr. de Op., Alg. Voraz, Alg. Kruskal (Recubr.)*

- A. *Binar.:* A. t.q. $\forall gr(no_i) \leq 2$ (solo hay hijo izdo y/o dcho).
 - A. B. Completo: $\forall gr(no_i) = \{0, 2\}$ (todos los niveles están llenos)
 - Delete. hoja o n. con 1 solo hijo: es sencillo (no ambigüo)
 - Delete. nodo con 2 hijos: para evitar ambigüedad: el n. derecho ocupa la posición. (y se *Rot.*)
 - Usados en cod. *Huffman.*
 - *Impl.:* ver *Struct*

14.7 Árbol binario de búsqueda

RESUMEN de lo que queda de Árboles:

- A. Binar. Busq: cumple que todas las Key de subArb. izdo son menores que dcho, al visitar en InOrder (I,R,D) *Recurs.* salen en orden, da Busq. en BigO($<n$)
- A. B. B. Balanceado: el peso de rama izda = dcha, mantenido gracias a rotaciones, da BigO(logn)
- A. B (de Boeing): tipo de A. B. B. Bal. m-Ario, pero que guarda Key y Dat en comparación con B+ (ver), da BigO(logn)
- A. B+: tipo de A. B. m-Ario donde m es *Masiv.* y cada Nodo solo guarda Keys o Ind. (no datos), para Busq. Vel. en *Sist. Arch.* y VSAM *Masiv.*
- Los *Ind.* de *BD* se hacen con A. B, B+ o *Hash*
- *Busq.* o Recorrido A.: *Alg.* que visita todos los nodos 1 sola vez. Ver Arbol ABCDEF de Fig. 14.11.
 - PreOrd.: F,B,A,D,C,E,G,I,H (Raiz, subarbol Izdo, subarbol Dcho): a) visita nodo actual, b) *Recurs.* recorres el subarbol Izdo. c) *Recurs.* recorres subarbol Dcho.
 - InO.: A,B,C,D,E,F,G,H,I (I,R,D): a) *Recurs.* recorres subarbol Izdo. b) visita nodo actual, c) *Recurs.* recorre subarbol Dcho. En Arbol Binario Busq. sale ordenado. (ver abajo).
 - PostO.: A,C,E,D,B,H,I,G,F (I,D,R): idem.
- *Arbol Binar. Busq.:* $k(\text{no}) \{ > k(\text{no}_{\text{left}}) \& < k(\text{no}_{\text{right}}) \}$ i.e. todos las Key o Id. de subArbol izdo son menores que los del dcho, i.e. la *Etiq.* de cada nodo interno es mayor que todas las claves del subárbol izquierdo del nodo respectivo y menor que las de su subárbol derecho. Ej: Fig. 14.11 tiene $n = 9$, $k(\text{raiz}) = 8$, $NNiv = 3$
 - *Busq.:* a) es $\text{BigO} < n = [\log n, n]$ donde $n = NNodos$. b) InOrder: las claves aparecen ordenadas!, Fig. 1,2,3,4,5,6,8,8 c) puede hacerse *Recurs.* o *Iter.*
 - *Insert.:* 1o Busq. binaria.
 - *Delet:* a) si 0 o 1 hijos como antes. b) si 2 hijos: sustituir por nodo hoja con clave a) menor de hijos derechos o b) mayor de hijos izdo.

14.8 Árbol binario de búsqueda autobalanceado

- *Rot. Arbol :* op., más común de **Balanceo**, que cambia la *Estr.* sin interferir con el *Ord.* de *Busq.* de los elementos. Los A. α, β, γ Fig. 14.11 observar, que si a *Nivel* más *Profund.* menos peso y que raiz es centro, ambos están balanceados. Ver *Matr., Algebr.*
- A. B. B. *Equil.* o Auto-Balanceado: para todo Nodo la profund. de rama izda = rama dcha \pm una Ud. i.e. mantiene Autom. su *Profund.* pequeña ante I/D, gracias a *Rot..* Junto con *Hash*, se usa para *Dic. Efic..*
 - *Busq.:* mejor que Hash en el peor de los casos ($\text{BigO}(\log n)$ en comparación con O(n)), pero tienen un peor rendimiento en el caso medio (O(log n) en comparación con O(1)).

- Ej: más famosos A. AVL and A. red-black.
- A. B (de Boeing o *Busq.*): A. auto-Balanceado pero no binario (sin límite de hijos, *m-Aria*). Permite *Busq. Efic. BigO(log n)*. Esto lo hace útil para *Sist. Arch.* y *BD Masiv.* (usado en *ISAM* de *IBM*).
 - A. B+: es una variación donde *m* es *Masiv.* y cada nodo solo tiene *Etiq.s (Ind. o Key)* y *Punt.* (sin datos!). Las hojas (con *Dat.??*) se encuentran en nivel bajo, *Enl.* entre si con una *Lista E..* Esto permite recuperación en rango con *Busq. Sec.* Usado en *Sist. Arch.* como NTFS (o EXT4) y en *SGBD* para *Busq. Masiv. Vel.* (con *Ind.* en BD). *IBM* lo introdujo para su *VSAM Hist. SO.*

14.9 Grafos

- *Graf.*: *Estr. Dat. Din. No Lin.* varias def. *DiGraf.* y *Pesos* .
- Tipos G.:
 - In/Conexo: *NN* bipartida. Estrella (2 triangulos) Fig.
 - DiG.: se define con *Tupla*. Hay sencillo o múltiple > 2 enlaces entre 2 no. *Hiper-graf.* Fig.
 - G. Pesado: *Probl. Karp Viajante Fig. (Rut. corta P vs Viajante NP).* *Flujo*
- Def:
 - Clásica: *Tupla*: v+a (Pares (*Ord.*)) con Nod. (elem. *Conj.*) y Arist. (*Rel.*)
 - *Lista de L. o L. Adyac.*: como def. matem. En L. están todos los nodos y en cada uno sus *Rel.* ((1,2,5),(3,4),(2) DiG.??). Se hace con *Punt.s*. El *Probl. Inv.* de las Rel. es no *Efic.??*
 - *Matr. Sparse* (dispersa): *Equiv.* a clásica, solo pares de enlace ((2,1),(1,2),(5,3)). Mas *Efic.* que *Matr.*. Ver F. *Hash, Ind. S. (Factor Block BD)*.
 - *Matr. (Array) Binar.* $G(i, j) = \{0, 1\}$
 - *Tip. Dat. Abstr.*: def. por *Usr*
- *Busq.* o Recorrido en G.:
 - B. 1o en Anchura o **BFS** (Breadth First Search): 1 se recorren los nodos a 1 arista de *Camino* del de salida, luego los de 2, etc.. Ver orden de recorrido en Fig. 14.11. *NP Hamilton* impide *Efic.??*
 - B. 1o en Profundidad o **DFS** (Depth First Search): se baja todo lo que puede y luego *Backtracking* si no hay más. Ver orden en Fig. 14.11 1,2,.. y *A,B,D,F,E,C,G* (1o escoge izda, si *Circ.* vuelve a *A* se descarta). Euler para ir descartando??. Ver *Backtracking*
 - B. *A** (*Alg. A estrella o star*): mezcla B/DFS

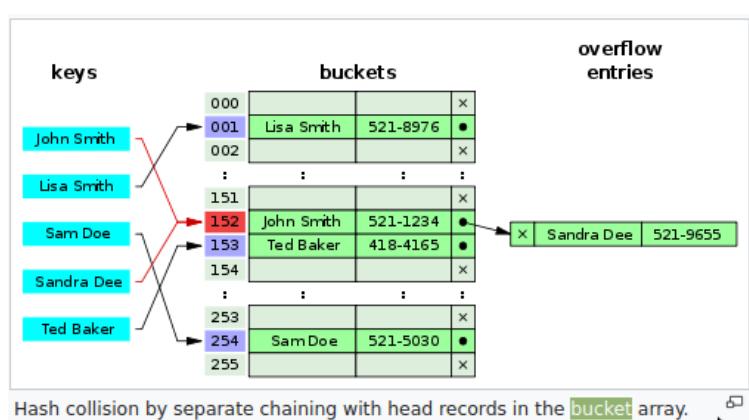
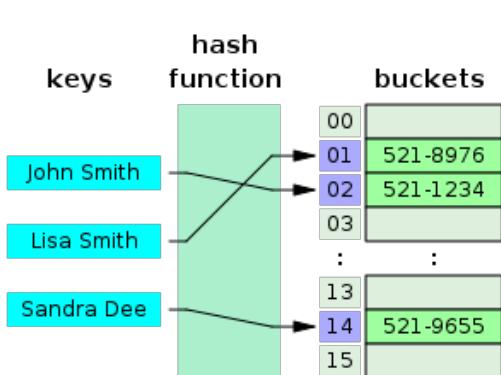
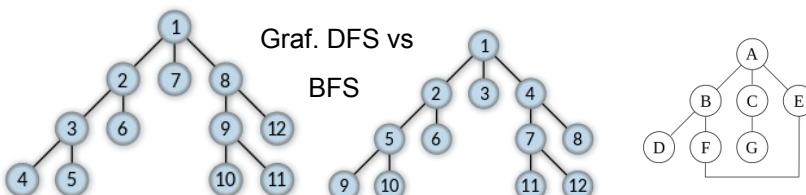
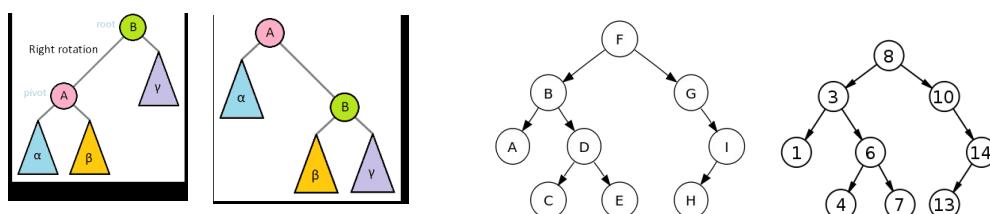
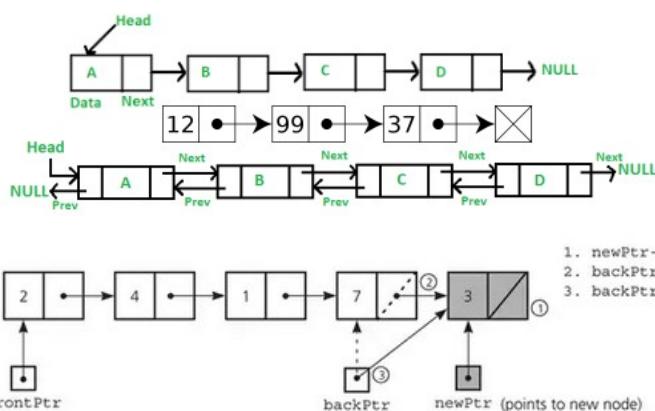
Ver Alg. *Fulkerson*

14.10 Tabla Hash

- Tabla **Hash**: es un tipo de Mem. Asoc. que junto con *Arbol B. B. autoB.* sirve para hacer *Dic. Efic..*.
 - Funcionamiento: dada la Clave (*Key* o *Etiq.*), encontrar el Valor en un *Dic.* desordenado es poco *Efic.* (*Vel. Acces. lenta*). Si en medio ponemos una *F. Hash* que Asoc. un N° *Ind.* (*Direc. Mem., Bucket, Slot o Cod. H.*). a cada Clave (*Index(Key)*) = $(\sum_i k_i) \pmod{N}$ con k_i valor *ASCII*) la *Busq.* es *Vel.* si *F.* lo es.
 - *F. H. o de Dispersión*: es *Vel.* en comput., *Modul.*, *Sparse*, y *PseudoAleat.* (no continua, o coativa). Ej. *SHA*. Otras son los plegamientos de dígitos (ver Regla 4 Últimos Digit. *Acces. Hash (SAM)*).
 - *Colis.* (no confundir con Dom. C. *Ethernet*): se produce cuando dos Claves van a la misma Direc. (por la *Correspond.* No Univoc. de modularidad). Si ocurre *Enl.* a Zona *Overflow* (como *SAM*). Si muchas C. *Migr.* a más grande. Ej. de *Vulner.* por *Colis.*: *MD5 (Cript.)*.
 - Factor de *Carg.*: $f = n/k$ donde k nº elem./buckets/slots o tamaño total de la tabla hash y n capacidad de tabla o nº entradas ocupadas en la tabla o nº preferido de entradas que se pueden *Insert.*ar antes de que se requiera un incremento en el tamaño de la estructura de datos subyacente (haya *Colis.??*). $f = 0.75$ es un buen **Compromiso**, más alto disminuye la *Sobrecarg.* de espacio pero aumentan el coste de *Busq.*. Más bajo tiene períodos de uso muy elevado (demanda) y una tasa de utilización baja.
 - *Impl.* en C: *Lista Enl.* por *Colis.* con *Punt.* según Fig. 31.10

Ver TPM (*Chip*), *Checksum*, *Punt.* Java, *Ind.*

14 – *12. Organización lógica de los datos. Estructuras dinámicas.



Chapter 15

*13. Ficheros. Tipos. Características. Organizaciones.

15.1 Introducción. Conclusión.

- Ej-motivador: ¿qué es más rápido Busc./Editar, una cinta de video o una *RAM*? Sol. $BigO(n \text{ vs } < t_{max})$ La *Ind.* permite buscar si desordenados pero con *Block* ordenados.
- Ej: ¿como Busq. rápido y crecer *BD*, *Sist. Arch.* o *Fich. Masiv.* de un *Mainframe*? Sol. *Arbol B+* (*VSAM*)
- *Hist. SO: Tarj. Perfor??*, F. Orient. *Registr.* de Long. Fija (CP/M [Kildall76] *Hist. Win*). *IBM OS/360*, met. *Acces. SAM*.
- Futuro: *Blockchain* y *Git*

15.2 Ficheros o archivos

- *Fich.* o *Arch.* : *Estr. Dat.* Externa. Conj. de Bytes (=Dat. de *Inform.*) con un *Id.* de *SO* (se Rel. Log. con otros, es un *Recurso*) *Almac.* en *Dispos.* de *MS*. Llevan una *Cabecera* de *Metadatos*:
 - *Atrib.*: a) ruta+nombre+*Exten.* (*Tip.*) para *Busq.*), b) u **Oculto** para *Segur.-Confid..*
 - Fecha creación (+ último Acces. o Modif.)
 - Tam. o *Esp.* en Bytes o *Bloq.*
 - *Permisos* (ver abajo).
 - *Tip.*: 7 en *POSIX/Unix* (ver abajo) (todo lo es)

Unif. por los 7. Ver *BD*, *Sist. Arch.*, *Admin. A. (Dir.)*, *Multimedia*, *Container*, *Pipe*, *Arch. Dispos.*, *Registr. Log*, *INode*, *F. Config.*

15.3 Características

- Caract. *Fich.*: a) *Dinam.*: tam. cambia b) *InDepend.*: R/W por cualquier Apl. c) Tamaño *Masiv.* » que *MP* Ej. *Model.* de *Aprend. Autom.* d) Factor de *Bloq.*: (ver *Ind.* en *SGBD*).

15.4 Los 7 tipos de ficheros

- *Tip. Fich. Unix* [Unix file types]: en *POSIX/Unix* son 7: regular, directory, symbolic link, FIFO special, block special, character special, and socket. Salen de *Comando ls -l*
 1. Regular:
 2. *Dir.:*
 3. *Enl. Simb. (Ln):*
 4. *Pipe:*
 5. *Arch. Dispos. Block:*
 6. *Arch. Dispos. Caract.:*
 7. *Socket.*
- Según Ejec.: En *Unix* Fich. son todo (*Descr. Arch.*) a) F. Regulares *Ejec.: Binar.* o *Script* c) F. Regulares *Multimedia: Texto* b) *Dir.* (tipo de F.) c) *Arch. Dispos.: de Bloq.* o Caracter

15.5 Tipos de ficheros

- Según *Tmp*:
 - F. *Tmp* (\neq *Persist.*): vida corta. a) F. Intermedios o de Result.: de *Compo. Proc.*, Ej: *Pipe* o F. *Impr.*. b) F. Maniobras: cuando info no entra en *MP* por *Masiv.* Ej: *Model. de Aprend. Autom., Mem. Virt. Swap*
 - F. *Persist.* o Permanente (\neq): vida larga, Inform para funcin. de Apl.
 - a) F. *Master*: almacena *Config.* o Dat. Modif. de una Apl. Ej: F. Clientes *Sucursal Banco*
 - b) F. Cte: Dat. que no cambian para consultas. Ej: ubicación de aulas en un centro
 - c) F. *Journal*: para volver a *Vers.* Ej: Hist. del *Master Git* o incidencias en Trenes. Ver *Acces. Secuenc. (SAM abajo)*

15.6 Tipos: Sistemas de archivos no y orientados a registros

- *Sist. Arch. No/Orientados a Registr.* ([Record Oriented File System] no confundir con *Arch. Dispos.*): dos tipos:
 - F. sin R.: *Secuenc. Bytes Unix* o MS-DOS
 - F. con R.: *Secuenc. Regist. con Metod. Acces. (SAM,.. abajo)*.

Otros usos: R. *Win*, *Fich. Log.*, R. *Fich. Texto=Linea, Journal*. R. *BD (Tupla)*, R. *Autent.* y *logcheck (Monit.)*, R. *Hardw. (CPU: R. uso Genral/Espec. Circ. Secuenc.: R. Desplaz.)*.

15.7 Sistemas de archivos no orientados a registros. Gestión de asignación y espacio libre

- *Descr. Arch.:* 3 nº de cada *Proc. POSIX (INode)*.
- *Asign. en Sist. Arch.:* ver RESUMEN *Gest. Mem.* 2x3 a) *Masc.* b) *Lista Enl.* y c) *Ind.*
1) *INode* (crecer Fich. *Dinam.*, Tabla *Descr. Arch.*).
2) *FAT (Tabla)*.
- *Esp. Libres:* *Control de Bloq.* con a, b y c de nuevo

15.8 Sistemas de archivos orientados a registros

- F. Orient. a *Registr.*: SO dice cuantos Bytes siguiente Registr.
 - R. Long. Fija y Var. (entre min. y max.): antiguos SO como CP/M (*Hist. Win*) y Fich. *Binar.* (el Byte, ver abajo).
 - R. Long. Indef.: la Apl. que lo Lee lo sabe por:
 - a) *Separ.*: lineas con *\n* y final con *EOF*.
 - b) *Cabecera: Container*
 - c) *Masc.:* *Map.* bits de presencia de Campos
 - Estr. *Log.*: similar a *BD* Fig. 15.14: *Jerarq. Mem.* bit, Byte, carácter, *Campo, Registr., Fich., BD*.
- Según contenido: como *Fich. C*
 - F. *Texto: Acces. Secuenc. caract. Registr. Long. Indef.* (ver *Separ.* arriba y *SAM* abajo). \nexists Rel. Dat. Mem. vs Dat. Soporte *Fisic.*
 - F. *Binar.:* *Acces. Direct. Registr. Long. Fija el Byte* (ver arriba). \exists Rel. Dat. Mem. vs Soporte. (son *Imag.*).

15.9 Organización de ficheros o métodos de acceso

- Metod. *Acces.* u Org. *Fich.* (no confundir con Mod. *Direc.* o Mod. *Acces. MP (Asoc.)*): es la forma de *Busq.* en los *Fich. Masiv.* de *Mainframe* que están orientados a *Registr.* (como *BD*). i.e. dado *Key Busq.* su *Registr.* Nacen con *SO IBM OS/360* en 1963 *Hist. SO*. Son F. o *APIs* y dependen de la *Estr.* del Soporte *Fisic..* Según Fig. 15.14 y [Prieto] hay 4:
 - A. *Secuenc.:* *Busq. dat. Lin.* Ej: Cinta Magn (*Disco*).
 - A. *Aleat.:* Busq. dat. Directamente. Ej: *Hash* o Direct-Access Storage Device (DASD) de *IBM*
 - A. *Ind.:* Busq. dat. usando un índice separado. Ej: ISAM y VSAM.
 - A. *Enl.* o *Cadena:* Ej: *Lista E.* o *Blockchain*
 - Otros: *Almac. Distrib. (Cluster)*.

Ver *Her.*, *Ambito*, *C. Acceso*, *CiberSegur.*, *Disponib.* *BD*, *Topol.* *Red*, *Firewall*, *Autent.*, *Recurso*, *Sec.*, *Crit.*, *Permiso*, *Vel.* *A. Mem.* (*Hash*), *RAM*, *DMA*, *Estr.* *Dat.*, *Transac.*, *Grup.* de *Usr*, *A. Aleat.*, *ACL* (*Autent.*). *Violación A.* (*Vulner.* *Overflow*). *Red A.* (*Internet*).

15.10 Organización secuencial

- *Secuenc.*: para ir a un *Registr.* debemos correr todos los anteriores. Tiene $\text{BigO}(n)$, i.e. no sirve *Busq. Binar.* $\text{BigO}(\log_2(n))$.
 - Ej: Cinta Magn (*Disco*) o *Texto* (ver *Separ.* arriba).
 - V: *Vel.* (si R/W *Bloq.* continuos). I: Lento (en caso contrario), deja huecos estilo *Petic.-Trim??*
 - *Actual.*: como en Control *Vers.* o *Journal* Fig. 15.14 (new *Master*=transact+Master)

15.11 Organización aleatoria o directa

- *Aleat.* (hoy) o Directo (antes, no confundir con *DMA*): como *RT*, podemos ir a cualquier *Registr.* en el mismo *Tmp* acotado. Es *Busq.* en *Tabla=Key+Registr.*
 - A. *Offset* (o Dir. Relativa): coincide la Secuenc. de Keys con la Sec. de *Direc.* Mem. salvo un Bias. En Fig. 15.14 el Bias es 0
Ej: *RAM*, *HDD* o *Disco CD* con Menú.
 - A. *Hash* (o Dir. Calculada o *F.*) para obtener la *Direc.* Mem. debemos pasar la Key por la *F. Hash*.
Ej: *Hash* ver abajo.
Inconv.: si *Hash* puede dar huecos y **Sinonimos** o **Colis.**

Es *Unif..* Ver *Fuente Inform.*, *Bosque A.* *Ln*, *Pasw.* (*Autent.*), *Prob.*, *Est.*, *PseudoAleat.* (*RSA*)

Ej. 11.4 **Hash Regla de los 4 Ultimos Digit.** (RUD): una empresa guarda la info de sus clientes en un Fich. con org. Dir. Rel., para ello sigue la **R4UD** del DNI, ej. la info del *DNI* = 27947512 se almacena en *direc.* = 7512. I. de dar huecos y Colis. como 28547512, 75937512,... Ej. 11.5 Igual pero con un org. Dir. Hash, para ello sigue la **regla del plegamiento en grupos de 3 digit.**, ej. la info del *DNI* = 27947512 en *direc.* 27 + 947 + 512 = 1486.

15.12 Organización secuencial en cadena. Blockchain

- Org. *Secuenc.* *EnCaden.* [Prieto]: *Fich.=Lista Enl.=conj.* *Dat.+Punt..*
 - V: *Dinam.* (*Insert.* flexible) sin reorganizar (como *ISAM*). Cumple *InDepend.* Log.-Fisic. (como *Pagin.*)??
I: No *Consult.* *Secuenc.*

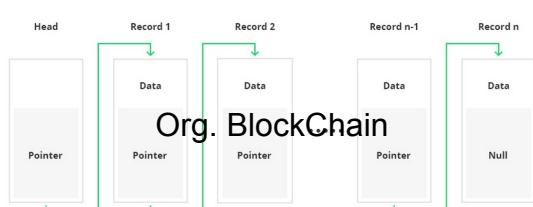
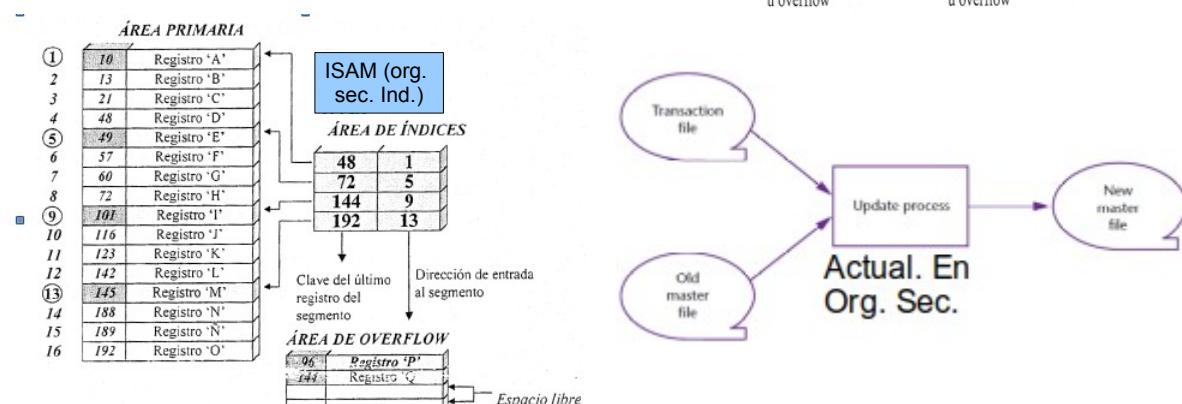
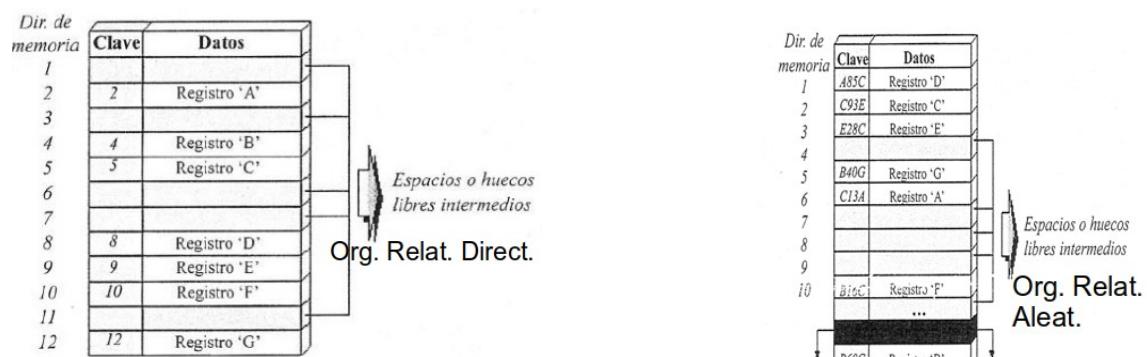
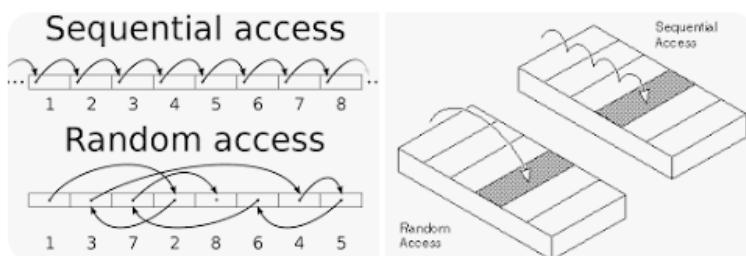
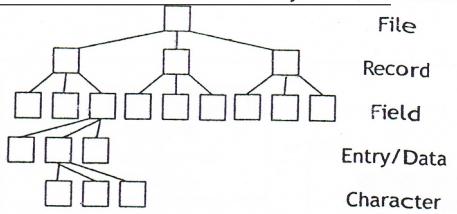
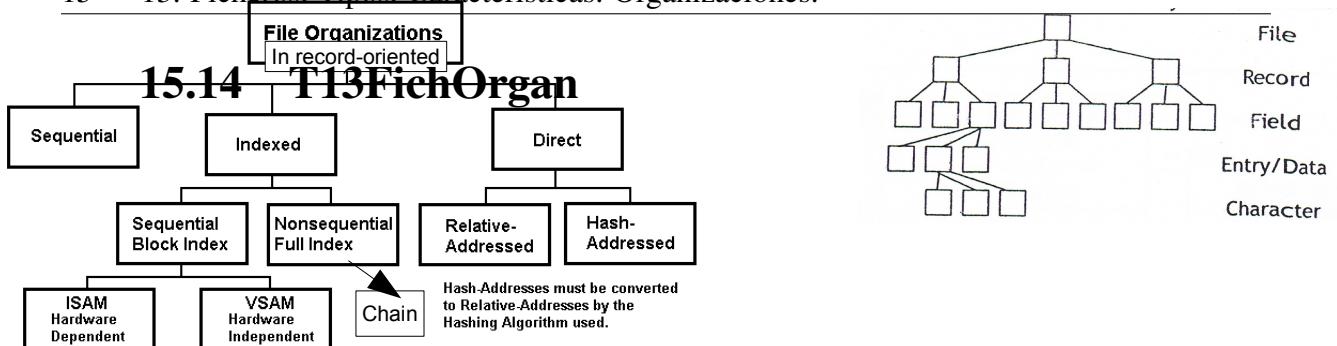
- Rel. con: *Block Chain* (solo comentarlo) e *ISAM* (Org. Secuenc. En*Caden. Ind.*)
- **Block Mem.** (block data storage, no confundir con *Bloq.*): *Ud. Atom.* o min. cantidad de Bytes que pueden transf. en una op. de *I/O* (a *RAM*, *MS* o *Perif.*). Para *HDD* suele ser de *512Bytes* (como *TCP*) y *Disco* opt. de *2KB* (ver)
 - *Caden. Block* (Blockchain): *Fich. Distrib.* o *BD Distrib.* (*Almac. Distrib.*) en cadena *Bloq.* como *Lista Enl.*, cada elem. aPunt. al siguiente. ∈ “mundo de los *Token*”. Ej: BitCoin, NFT (Smart Contract) en *Certif.* y *Firm.*). En *Centro Dat.* poco *Ecolog..* Basada en *P2P*

Es *Unif.*. Ver Factor B. (*Ind. SGBD*). SAN (*Almac. Distrib.*), *PCB*, *Fragm.*, *Disco*, *Flash*, L. *Interpr.* L. *Visual*, Progr. *Estr.*, *Estr. Control*, Tabla B. Ind. *INode Estr. Control*

15.13 Organización indexada: SAM

- V/*ISAM* (ver)

15 – *13. Ficheros. Tipos. Características. Organizaciones.



Chapter 16

*14. Utilización de ficheros según su organización.

16.1 Métricas de uso de ficheros

- *Esp.* o Volumen: $V = NRT * LR$ nº Registr. por Long. (en Bytes o Caracteres) Media de Registr.
- Segundo *Op. BD*: Fich. Estat. o Estable (T12) si Tasa *Volatil* baja, si no *Dinam..* Para cada *Metr.*, Frecuencia (en vez de Tasa) si promediado en periodo de tiempo:
 - Crecimiento: $C = \frac{NRA}{NRT} (\%)$ nº Reg. que Aumentan (*Insert.*) en cada tratamiento. Es Ley Logar. Ver *Escal..*
 - Actividad: $A = \frac{NRC}{NRT} (\%)$ nº Reg. *Consult.*
 - *Volatil*: $V = \frac{NRAB}{NRT} (\%)$ nº Reg. Alta o Baja (*Elimin.*)

16.2 Organización de ficheros o métodos de acceso

- RESUMIR T13: este T14 es como T13 solo que aquí se detallan como se realizan las *Op. BD* sobre cada tipo Fich. (*Actual.* y *Consult.*).
- Metod. Acces. u Org. *Fich.*: para Fich. Orient. a *Registr.* Fig. 15.14
 - *Secuenc.*
 - *Aleat.*
 - *Secuenc. EnCaden.*: *Block*
 - *Ind.*: I/VSAM

16.3 Índices, factor de bloque y búsquedas masivas.

- *Ind.* ice o Indexar (similar a *Enum.* o *Etiq.* o *Key*): componente que permite *Ord.* los datos y *Opt. Consult.* (por ser *SQL Decl.*). Es *Filos. MATLAB*. Se crean y borran en T. Ejec., rápido si tablas al menos en 3FN. Ej: *SQL*

```
CREATE INDEX my_index ON my_table (id) INCLUDE (name);
```

- *Ind. Sparse: Tabla Ord.* de *ClPr+Punt.* al *Block MS* de cada *Registr.* Mejora enormemente *Vel.* ver Factor *Block* (ver abajo).
- I. populares son: *Arbol Balanc.*, A. *B+* y *Hash*, (*Index VSAM*)

Ver *Enumer.*, Mod. *Direc.* Indexado. *INode*, *Array*, *Hash*, I. TIOBE (*Popular*). Search Engine Index (PageRank, Naveg.).

- Factor de *Block* (Blocking F. o F. Bloqueo): $FB = Floor(B/R) = Floor(512/5) = 100$, donde *B* y *R* son los tamaños en Byte de Bloq. y Registr. i.e *Metr.* nº de *Registr.* Log. (de *SGBD* o *Fich.*) contenidos en un Bloq. *Mem. Fisic.* (*Arch. Dispos.*), cuanto mayor sea el nº de *Acces.* será menor pero cada *Acces.* será menos *Vel.*. <https://stackoverflow.com/questions/15859070/blocking-factor-in-a-dbms>. Ver *ISAM*

- 1) Sea una *Tabla* de Num. *Registr.* $NR = 10000000 = 10MReg.$ cad uno de Tamaño $TR = 80By/Reg.$ (incluyendo su *ClPr*), si el Tam. *Block* es $TB = 5000By/Bl.$ $\Rightarrow FB = 5000/80 = 62Reg./Bl.$ luego Nº *Acces.* para acceder o guardar toda la BD es $NAc. = 10M/62 = 161000Bl.$, esto es malo (*Cuello Botella*) pues si queremos Busq. un Regist. dada 1 *ClPr.* (ej. info de una persona dado su DNI) son muchas Petic. a MS que es muy lenta comparado con *CPU* o *MP*.
- 2) Sea ahora una 2^a Tabla Ind. Sparse con Tamaño *ClPr+Punt*=6 + 4 = 10By/Reg. $\Rightarrow FB = 5000/10 = 500Reg./Bl. = 500Cl - PuntBl./Bl.$ i.e. 500 parejas *Cl-Punt.AIBl.* por *Bl.*, el nº de *Acces.* a esta tabla es $NAc. = 161000/500 = 323Bl.$, mucho menor que la anterior. Es más si consideramos que está *Ord.* el nº de *Acces.* por Busq. Binaria es como mucho $NAc. = \log_2(327) = 9$ ¡hiperreducción!

16.4 Organización indexada: tipos (SAM)

- **SAM** (Storage/Sequential Acces. Method): *Estr. Dat.* (junto a *Arbol B+* y *Hash*) forma famosa en la que *SGBD* y *Fich.* guardan sus *Da.*
Para *Fich.* orientados a *Registr.* (*Busq. Vel.*, *Crec.* *Dinam....*)

 - ISAM (*Ind. Sequential A. M.*): *Depend.* del *Hardw..* Busq. ej. abajo y Fig. 15.14:
1º) en *Tabla Ind.* (*Secuenc. lento*) saca límites del *Block*.
2º) Busq. en *Block. Ord..*
3º) si no en *Overflow* (como *Hash*) que está des*Ord.. Opt. Consult. Topos*
 - VSAM (*Virt. Storage A. M.*): Independ del Hardw. (**Ind. Log.-Fisic.**). ISAM usa *Arbol B* (sin +) mientras VSAM *B+*.

Ver Mod. *Direc.*, *Ord.*, *Sist. Arch.*, Ver *CODE2*, *Manej.* Excep. *Web stackoverflow, Tip., O. Enteros (scanf C), Buffer O. (Vulner. Segmentfault Null)*

16.5 ISAM

- *Insert./Eliminar:* es Marcar (*Trim*) y añadir en *Overflow*. Inconv.: *Ord.* periódicamente *Fich.*

- *Overflow*: en ISAM es el área para reescribir un Fich porque se *Actual.* por *Trim*.
- Mejoras:
 - a) Tablas **Cascada** (Busq. *Binar.=CSS*).
 - b) Poner Tabl. Ind. en *MP*.
 - c) VSAM (*Arbol B+*).
- Opt. Factores de Bloqueo (FB, *Ind.*): nº de Acces. min. si nº Registr. por Block cumple $N_t = \sqrt{RB_r/B_i}$ donde R (nº total Registr.) $B_{i,r}$ son los FB de Tabl. *Ind.* y Registr.

Ej. Fig. 15.14+[Wiki], se quiere acceder al *Key* = 60, entonces se Busq. Secuencialmente (lenta, no *Ord.*) en Tabl. Ind su *Block* que es $(Key_{max}, Dir_{ini}) = (72, 5)$ y se empieza a Busc. Secuenc. desde *Dir* = 5 hasta que se encuentra que el valor del Reg. es *Reg*(60) = *G*. Si no se iría a *Overflow*. Es Opt. Consult. Topos

16.6 VSAM

- VSAM: reemplazó al ISAM. Lo lleva IBM en su SO z/OS y su SGBD DB2. Tiene 4 formas de Estr. los DS (Data Set, Conj. Dat.):
 - RRDS (Relative-Record): Relat. Aleat. (rápido), como Fig. 15.14
 - ESDS (Entry-Sequenced): Relat. Secuenc. (lento)
 - KSDS (*Key*): usa *Arbol B+* y es un Indexed VSAM (I-VSAM).
 - LDS (*Lin.*):

16.7 Otros detalles de ISAM

RESUMEN: ej. de VSAM para Busq. Vel. en SGBD. ESDS=RRDS y casi= Relat. Aleat., permite ambos Acces. Direct. y Secuenc.. (es ventaja pues las Consult. Secuenc. son frecuentes) pero dejan huecos y Consult. Lentas.

ISAM usa Área *Overflow* (como Hash) con *Arbol B* (tiene las ventajas de Acces. Secuenc. y Relat. de los anteriores) pero mejorado por VSAM que usa *Arbol B+* en KSDS (*Key*).

- ISAM (Org. Secuenc. EnCaden. Ind., no confundir con KSDS abajo): Busq. Vel. por Punt. pero mayor Esp. ocupado. Tiende a Fich. Secuenc. si las elimin. no se reOrd. (porque Área *Overflow* crece).
 - *Ind.exada Secuenc.* (ISAM): el Fich. tiene Tabla Ind. (pequeña, solo Acces. Secuenc. no Busq. *Binar.*) y Tabla Registr. (grande y Ord., en zona *Overflow* están desord.)
 - V: de Secuenc.+Relativa (como Segment. Pagin.=Busq. por Bloq.). Útil para combinar Consultas a Registr. concretos y el Proc. es Secuenc. en todo el Fich.

Mejoras de Vel. Busq.:

- **ISAM en Cadena**: cada Registr. además tiene un campo Punt.. Si se hacen muchas Actual. (borrados Trim o Inserc. en Overflow) se relentiza Busq. y es necesario Reorganizar de nuevo la Tabl.
- **VSAM**.

16.8 Otros detalles de VSAM

- *VSAM*:
 - (ESDS) entry-sequenced y (RRDS) relative record =Relat. Directa o *Secuenc.*: un *Fich.* es una *Tabla*: *Direc.* *Mem.+Key* (*Key o Etiqu.*) + *Dat.*. i.e. para acceder a un registro concreto hay que buscar en todos los registros secuencialmente hasta localizarlo.
V: R/W simultaneamente, *Acces.* Direct. inmediato (dada *Key*) o Secuenc. (desde el 1er Registr.). I: Huecos (las *Keys* pueden no ser continuas), *Colis.* (puede \exists mas de 1 Registr. con misma *Key*), Lento en *Busq.* (implica recorrer *Direc.* vacias)
 - Relat. (*Offset Aleat.*): *Keys Alfanum.* no coinciden con *Ord.* de *Direc.* V: *Acces.* inmediato (dada *Key*), no necesario *Ord.* Registr., R W simultaneas, posible *Acces.* Secuenc. I: Huechos (muchos), Lento. Requiere F. Convers. *Keys* que evite *Colis.* (como *Hash*), puede dar **Sinonimos=Colis.**
 - (KSDS) key-sequenced=Org. *Ind.exada* (no confundir con ISAM aunque *Actual.* usa la *Estr. Dat.* de *Arbol B+*). *Tabla Ind.* para *Acces. Aleat.* (sin *Secuenc.* o pasar por todos). V: *Acces. Directo* I: Predecir tamaño y mantener *Tabla Ord.* (por *Key*)
 - (LDS) linear:

16.9 T14FichOrganUso

Ver Fig. 15.14 del T13FichOrgan

VER T13FichOrgan

Chapter 17

+15. Sistemas Operativos. Componentes. Estructuras. Funciones. Tipos (9FebAngela)

17.1 Introducción. Conclusión

- Ej-motivador *Arduino*: ¿podriamos ponerle varios programas?
- Historia: MIT, +, Tolvard, Gate *Hist. SO Hist. Linux, Hist. Win THEOS (Multitar., Semaforo y Pagin.) OS/360 IBM*
- Futuro: *Naveg. SO, Virt. SO (Kubernetes), RT Embeb.*

17.2 Arquitectura, recursos y cargas

- Arq. *Neumann* y *Harvard*
- *Recurso Compart.*: elem. de un *Sist.* útil, limitado, competido o *Acces.* por varios individuos. Ej. 3 *Probl.* R. Compart. *TAP*. Es *Unif.*. Ver *Interfer.*, *Colis.*, *Red Bus*), *Efic.*, *Traf.*, *Transac.*. Normalmente T. *Ejec.* (*CPU*) y *Mem.*), *Pipeline*, *Servic.* *Dir.*, *URL Web*, *Petic.*, *Esper.*, *Prod.*
- *Carg.* :
 - *Equil.* C.: *Metr.* de Balanceo. Ej: *Switch*, *QoS* *T. Resp.* Medio en *Plan.* *Proc.* (en ambiente *MultiProgr. Multitar.*). Ver *ASimetra*, *Arbol Rot.*, *Servid.* *Apl.* (*Applet*)
 - Progr. del *Kernel* (se inicia con arranque) que pasa programas (*Gest. Mem.*) y *Bibl.* (*Enl.*) a *MP*. Si mala *Gest. Mem.*, puede dar retrasos de C. Ver *Asign.*, *Mem.*, *DRAM*, *CPU*, *Instr.* C. *CODE2*, *Equil. C.* (*Multiproc.* y *Plan.* *Proc.*)
 - *Cambio Ctx*: perdida de t. en la C.
 - *Comando top,ps*: load average: 0,28, 0,56, 0,73 (1 min, 10 min y 100 min??). Para *RT*

Ver C. Útil (payload, *PDUD*., *Sobrecarg.* (*Polimorf.*), *Cuota*, E. *Flujo*. *Kubernetes*

17.3 Sistemas Operativos y funciones

- **SO** : *Filos.* de **Maq. Extend.**: Progr. que Gest. otros progr. o Progr. *Virt.* que toma Apl. como I. los Proc. en el Hardw.
Gest. *Acces.* de programas, a unos *Recursos* limitados (*Hardw.*) y facilita al **usr/s** la ejecución de sus programas
Cuida su *Segur.* (sirve de base de lanzamiento).
Gest. *Transpar.*: hace que el programador o usr no se preocupe de los detalles o particularidades (similar una *Maq. Virt.*). Ej: programador de *Perif.* o usr de *Plan. Concurr. Proc. y Mem..*
T. Resp. bajo y *Vel.* alta
Todo SO debe permitir:

- Las 3+2 de *Arq. Kernel*:
 -) *Plan. Proc. (PCB PID)*: CPU, crea, planif, mata, comunica y sincroniza.
 -) *Gest. Mem. y Mem. Virt.* (TLB con trozos de Tabla *Pagin.*): *MP, Asign. y Librerar*
 -) *IPC*: comun. *Proc.*
 -) *Fich. (INode o Descr. Arch.)*: MS, *Sist. Arch.*
 -) *Perif. (Arch. Dispos.)*: *Driver, Manej. de Interr.* Son *Unif.* todo son *Fich.*
- 4 *Tablas* básicas (de *Unix*): en paréntesis *Estr. Dat.* con Info de todas salvo *IPC* (creo)

Ver *Econom.*, SO *RT*

- F. SO:
 - F. principales:
 - Shell: CLI o GUI* (Escritorios)
 - Config.* el propio Softw. SO (ej. modificar *Registr. Win*).
 - Instal.* otros Softw. (Apl.)
 - Segur.:* *Permisos y Grup., Acces.* a cuentas
 - Monit.:* Estad. de uso
 - TAP:* Gest. *Red y Protoc.*
 - Otros F.:
 - Det. *Err.* de Apl. o *Fich.*

17.4 Tipos 1: lotes, multi(tarea-usr-proc)

- *Proc. Lote* s (Batch Processing): *Secuenc.* de jobs que se pone en un *Script*. Ej. *Render.*
 - Parece puede entrar otro si uno espera una *I/O* [Wiki Multitask]

El usuario solo hace el esfuerzo de mandar un trabajo (job) partido en lotes. El SO se encarga de ir enviando los lotes a horas determinadas y cuando el hardware esté disponible. Un lote se ejecuta sin intervención del usuario. Se diferencia de T. *Compart.* en que este no permite ejecutar varios trabajos a la vez. . El nombre viene de las Formas de Producción: por job, batch (pintura de más clara a oscura para evitar limpiar) o flow. Ver *Cluster PBS, Bash*

- Por nº de procesos/tareas:
 - a) Monotarea: \exists_1 programa en CPU. Ej: *Lotes* (*Secuenc.* y cierra al *Usr*)?? Ej: CP/M *Hist. Win*
 - b) **Multitar.** (Multitask, no confundir con *Multiproc.* ni *MultiProgr.*): varios *Proc.* en 1 (o varias *CPUs*) *Ejec.. Concurr.* sin Sensación de Compart. *Recurso (Transpar.)* para aprovecharlos mejor (contrario a *Arduino*). Gracias a *Plan. Proc., Mem. Virt., etc..*
 - **THE** Multiprogramming System (o THE OS o Technische Hogeschool Eindhoven, no confundir con THEOS de IBM): *Dijkstra68* de los 1º con *Multitar.* (pero no *MultiProgr.*), *Semaforos* y *Pagin. Hist. SO.*
 - Multi*Prog.ramación*: lo asumido en ejercicios de *Plan. Proc.*, 1 sola CPU pero varios *Proc.* en *MP* (mejorado con *Mem. Virt.*) ejecut. *Concurr.*. Se ejecuta 1 sola tarea hasta que *Plan. Proc.* dice que le toca a otro o tiene que esperar un *Perif.* (producido por *Cambio Ctx*). Interesa **Equil. Carg.**
 - T. *Compart.:* ver abajo
 - **Gr. M.** (nº de *Proc.* ejecutándose simultan. gracias a *Mem. Virt.*). La Ejec. *Paral.* real ocurre con más de una CPU
 - Todos los SO modernos permiten M. Ej: WinXP, ServerUnix, MacOS, UnixAndroid, Novel

Ver *Spooling*

- Por nº de *Usr*: clasificación decayente por SO Monopuesto vs en Red (*SO Servid., NOS*). Monousr (MS-DOS, Android), Multi*Usr* (Compart. *Recurso HD* y E/S, ej. *Unix*, Novell, Win-NT-Server).
- Por nº *Proc.:* que pueden sacar partido.
 - a) Monoproc.: Dos, Win3.
 - b) *Multiproc.:*
 - *Simetr.* o *Equil.:* todas las tareas usan todos los procesadores ver Fig. 17.11 Tarea1-Procj completo.
 - Asimetr.: el SO reparte las tareas a cada proces. según vayan llenandose las CPUs. Fig. 17.11 (la Tarea2 usa parte del Pr1 y Pr2). Pueden quedar Pr. sin usar Puede haber un Proc. *Master Multiproc.*

Ver *Intel*

17.5 Tipos 2: RT, Red

- Por *T. Resp.:*
 - a) T. *RT* (T acotado en *Cambio Ctx* y misma sección Cod.) para *Embeb.*
 - b) T. *Compart.* (Time Sharing): Sensac. (*Transpar.*) de *Multitar.* por *Plan. Proc.* como Multiprogram. (ver) pero con *Proc.* de varios *Usr* trabajando *Concurr.*. Aprovecha los recursos mejor que *Proc. Lotes*.
- Por *Distrib. Red* (*Servic.* ofrecido):
 - a) Centralizados: usr conectados por Terminales (*Shell*) (sin RAM ni CPU) a un *Mainframe*.

- b) Monopuesto: o de Escritorio (portatil) Ej: Win10.
- c) En red (*SO Servid. NOS*): *Servic. Dir.* de ordenadores, *Admin.* eficientemente sus *Recursos*
- d) *Distrib.*: *Servic. Dir.* o *Grid-Engine (Solaris-mc-???)*. El usr percibe el SO como uno solo *Transpar.* de tareas entre procesadores de un mismo equipo o diferentes equipos.

17.6 Componentes: arquitectura de sistemas informáticos

- *Arq. Sist. Informat.*: la referente es la de la Fig. 17.11 (similar a Arq. *Tanenbaum de I/O*):
Usr,
Apl.
(*API*,)
Bibl.,
(*ABI*)
SO
(*ISC*, *Instr.*)
Hardw.. Entre medias están las *API*, *ABI*. y *ISC* (*Instr.*).
- **API** (Application Programming Interfac, I. de Programación de Aplicación, no confundir con ABI ver abajo, ni *Driver???*) es un tipo de *Interf.* para que se *Comunic.* dos *Modul.* o *Compo.* de un Progr. o dos Apl. o dos Comput. Hay de varios tipos: Local (de SO, *POSIX*), Web (de Internet), *Bibl.*, microservicios, etc. Ej: *Bibl. PortAudio*.
 - *ABI* (Application *Binar. Interf.*): no confundir con API que es a nivel de Cod. *Fuente* (alto nivel, en *Arq. Sist. Inform.* está entre *Apl.* y *Bibl.*) y *ABI* a nivel de Cod. *Binar.* (bajo nivel, entre SO y Bibl. ver Fig. 17.11). Aparece en sist. *Embeb.*

Ver CUDA *GPU*, *Middlew.*, *WebUSB*, *Conect. BD*

17.7 Componentes: Arquitectura de los SO

- *Arq. SO*: está *Modul.* en *Capas* Fig. 17.11. Según Arq. Ref. son:
 - *Shell*:
 - *Servic.* o *Daemon* ver abajo.
- *Kernel*: abajo

17.7.1 Shell: GUI y CLI

- **Shell** (Interprete de Comandos): Tiene dos *Interf.* (Fig. 17.11): *GUI* o *CLI*. Es una de las *Capas* del *SO*, es un *Softw. Interf.* que comunica los *Servic.* del *SO* al *Usr.* Nombre por ser la Capa más externa del *SO*.
- **Command Prompt**: CLI de *SO* como *Win* (i.e. el *CMD*).

- **CLI** : forma de *Shell* (Command Line Interface o de texto, permite acceder a todas la Admin. del SO mientras que *GUI* no suele. Ej: *Bash*, o *SSH*. *GNU*, *CISCO IOS*).
- *GUI*: no toda *Config.*, X11 SSH, GNOME (GNU) o KDE Fig. 17.11

Ver *SSH*

Explica el *Flujo de Inform.* de Fig. 17.11: las *Secuenc.* de *Comandos* son *Pars.* por el *CLI* del *Shell* y pasadas luego al *Kernel*

17.7.2 Servicios o Daemon

- **Daemon** (diimon o *Servic.* o Progr. Resisd.): *Proc.* en 2º plano (*Back*) o hijo creados por *Fork* no controlado por *Usr*.
Nombré de Fernando Corbató (*Hist. Linux*).
Ej. Cuando *Servid. Web* XAMPP (*Stack Soluc.*) lanza una Apl., este lo hace como *Usr D.* del *EtcPasswd*.
Ej *Linux*: *Comando cron* PulseAudio, D-Bus, NetworkManager, Qt. Ver 17.11.
Ej. **systemd**: *Proc.* Id=1 (el 1º que lanza todo en *Linux*
 - *Servic.* en *Win NT*: son los *Daemon* (hora, impr., red...).
 - QUITAR: Servicio: interfaz de APIs para abstraer programas del hardware. Ej: *Admin. Arch.* o *MS, Plan.* Proc. (lanzar-parar, *Fork-wait*), Gest. *I/O*. Otros: Proc. *Lotes*, **Config.** y *Monit.* del SO. *Pila Protoc.*, *Middlew*.

Ver *Spooling*. MDaemon (*Email*), *TCP Wrapper* (ACL Autent.), *RT*.

17.7.3 Kernel e IPC

- **Kernel** : *Interf.* con hardware que gestiona: recursos (CPU, RAM,...,) e *Interr.* (E/S). Ej: *Gest. Mem..* (Asign.), *Carg.. Plan.* Proc.. Sus 3+2 F. ver MicroArq. *Kernel* Ej: *Linux K.* incluye *ALSA* (*Bibl.* PortAudio)
- **IPC** (InterProcess Communication, no confundir con el IPC (Instr. *PS* de *Superescalar*): una de las 3 F. básicas de MicroArq. *Kernel* siendo la forma que tienen el *SO* de hacer *Concurr.* (*Fork+Sincr.*).
Se basa en Arq. *Client.* que hace *Petic.* (A/Sincr.) al *Servid.*

Las formas más comunes de los SO con *POSIX* son:

- *Fich.*: todos los SO (incluso sin *POSIX*)
- *Signal*:
- Las de *Concurr.*: Mem. *Compart.* (*Semaforo*, *Monit.*, *Mutex Excl.*) o Paso *Mens.* (*Pipe=Buffer*)
- *Socket: TCP/UDP*
- En *Red*: *Progr. Compo.* (*CORBA*)
- *Llam. a Proced.* (*SigHandler Manej.???*)
- Otros: *IRP??* (*Petic. Win*).

Es un *Unif..*

17.8 Arquitectura según Kernel

- **Arq. Kernel**: distinguimos 3 según peso del *Kernel*
- **Monolit.**: colección de muchas *F. Subrut.* en único *Bloq.* programa donde cada vez que se cambia algo hay que re*Compil.*. Sin Progr. *Modul.??* salvo los nuevos *Drivers* de *Perif.* que se añaden como *Modul.*
Ventaja: respuesta *Vel..* Desventaja: Su *Depur.* y mejora es compleja si muy grande. Se ejecuta en modo Kernel en un único espacio de direcciones. Ej: Importante *Linux/Unix* y *DOS*
 - *SO *Capas: Jerarq.* la capa superior solo usa *Servic.* de la inferior. Es el extremo contrario a Monol.
- **MicroKernel**: su kernel hace lo básico gracias al *IPC* que reduce sus funcionalidades al mínimo que son: a) *IPC* b) *Mem. Virt.* (*Gest. Mem.*) y c) *Plan.* Proc. Fig. 17.11. Las otras d) *I/O* las hace el *Usr* ⊃ e) *Fich.* (*MS*) Ej: *QNX* y *Minix*
 - Mod. *Client.-Servid.*: caso de Micro *Kernel* basada en *IPC*, que diferencia entre procesos cliente (usan servicios). Se comunican mediante Paso de Mensajes.
Ventaja: flexible ya que cada serv. hace una tarea concreta y pequeña facilitando la *Ing. Softw.* (desarrollo y depuración).
- **Kernel Hibrido**: combina V de monol. (respuesta eficaz) y microkernel (modularidad y robustez). Kernel hibr=microkerenel+driver+*IPC*. Ej: los actuales *Win*, o *Mac*. Fig. 17.11.
- **Maq. Virt.** (no confundir con *Virt. SO*): softw que usa softw (la MV y no hardw) para correr (*Ejec.*) un programa (*Virt.* o *Simul.*ado). El SO corre en una MV (máquina huésped software) que corre en un hardware (m. anfitrión). Podemos tener varios SO o PCs que se comunican. Distinguimos 2 tipos:
 - Requiere instalar GuestAdditions (para copiar pegar), *NAT* para usar red del anfitrión
 - de *SO*: Ej: *MVWare*, *VirtualBox* (de *Oracle*) *Cywing*
 - de *Proc.*: Ej: MV de *Java* **JVM** (le da *Portab.* ej: *Servid./Client. Videojuego.jar* corre en cualquier SO) o la MV *Android*-runtime
 - Exokernel: caso de MV desarrollado en el MIT. Mejora la MV al reducir al max. el uso de abstracción para acceder al hardw. Ej: EXOS del MIT o Nemesis
 - *Virt. SO: Container Kubernetes*

Sirve para *Naveg.* de forma *Segur.* por la *DeepWeb*. Con *Persist.* retomamos *Est.*

17.9 Evolución histórica

- **Hist. SO** : 5 generaciones de SO en lugar de 4 para coincidir con las de *Hist. A..*
 - 1^a 40-60: ENIAC no tiene SO
 - 2^a 60-65: Proc. *Lotes*. SO EXEC-I en UNIVAC

- 3^a 65-: Monousuarios.
- 4^a 65-80: Multi/*Usr*, tarea (no *GUI*).
- 5^a 80-20: GUI, PCS.

Ver *Hist. Linux*, *Hist. Win*

- *Hist. Linux* (*Hist. SO*):
 - CTSS [MIT61] (1º en tener T. *Compart.* y compatible con Proc. *Lotes*),
 - **Multics** [Fernando Corbató65 en *Bell* y MIT]: influye mucho a *Thompson* y resto, *Multiplex.*, *Arq. Kernel* Monolit. Fernando creó: *Daemon*
 - UNICS/Unix [*Thompson*69 en *Bell* y *Ritchie* (ver C)] (Uniplexed+POSIX, en C fácil portable, ver ARPANET), MINIX (réplica de Unix en libro educativo) [*Tanenbaum*87].
 - *Linux* [Tolvards91] (kernel basado en libro), *GNU-Linux* [Stallman] (añade Apl. *GNU* sobre el *Kernel*). Luego .deb y .rpm (ver L.)
- *Hist. Win* : MS-DOS [Gate&Allen75] (*Win* Apl. aparte), CP/M [Kildall76] (monotar y monousr *Multitar.*, ver *Fich. Registr.* Long. Fija), IBM-PC-DOS81 (con W1.0, **Acuerdo 82**, x86). Win 1.0 en [Gate85] (*CLI* + programa aparte de ventanas), W2.0, W3.0 en 90 y W3.1 NT Advanced Server en 93 (*SO Servid.*). W95 (1º que arranca con ventanas *GUI*, Escritorio+BarraTarea). Luego los Win NT uno bueno (XP, 7, 10), otro malo (Millennium, 8, 11).

17.10 SO actuales

- *SO Populares Client.* o Monopuesto (*Softw. Red.*): buscar *Win*, *Linux* para completar info.
- *Win* de *Microsoft* (*IDE VisualStudio/Basic/.NET*): **Win NT** es la familia desde 2001 que incluye Vista, 7, 8, 10, ahora 11 *Hist. Win*.
 - Ofrece la mayor gama de Apl.
 - *Arq. Kernel* MicroKernel. Tiene un Prompt (CMD) (PowerShell)
 - Win11: mejor que Win10 (rendimiento, uso, soporte para Android y Microsoft-Store), pero orientado al hardw. Por el *Chip TPM2.0*
 - *Registr. W.*: *BD* o *Arbol* de Direct. que almacena ajustes de *Config.*
 - Caract.: *Pagin.*
 - Otros: Win Server (*SO Servid.*), Win IoT o Embeb.
 - IRP (*I/O Petic.* Paq.), Plug and Play,
Cola Multicol. (EDF) *Plan. Prior. Dinam.*
Sist. Arch. NTFS

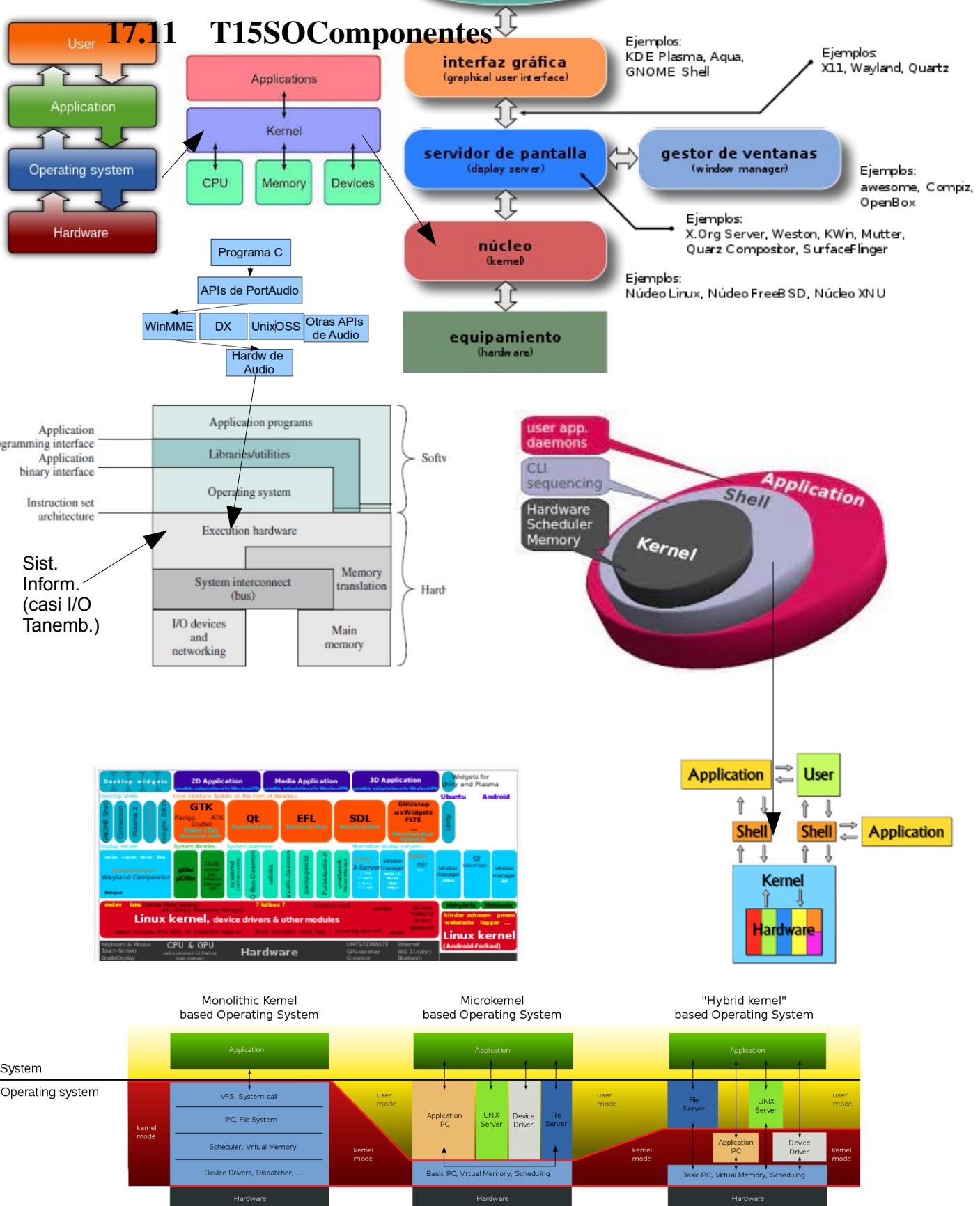
Ver *Hist. Win*, IPX/SPX NetBEUI

- *Linux* -GNU: *Arq. Kernel* Monolit.

- El *Libre* por excelencia. Mayor potencia, estabilidad, *Segur.* y modificable. Ojo: a veces no se puede copiar por contener Softw. de no Cod. Abierto.
- .deb (Debian/Ubuntu) y .rpm (Fedora/openSUSE): mismo *Kernel*
- **GNU** : [Stallman83], acrónimo recursivos GNU's Not Unix. Destaca por crear: *GCC*, *Bash (CLI)*. Ver *Libre*, *GZIP*, *GRUB (BIOS)*
- **Ubuntu** : la más *Actual.* es la 22.04. Tiene 3 ediciones: U. Desktop, U. Server *SO Servid.* y *Core* (para *IoT* y *Robot*)
- Escritorios: KDE, Gnome KDE (*Config.*).
- **Unix** : en *Bell* por *Thompson*69 (B y muchos, ver *Bell*) y *Ritchie*72 inventa C para añadirle F, (multitarea/usr y portable, *Hist. Linux*). Derivados: *Linux*, *FreeBSD*, *Solaris*.
Solaris
En *SO Servid.* y *Estac. Trabajo*. Son compatibles con *POSIX* (ver *Signal*, *INode*).
Los 7 Fich.
 - **Guadalinex**: 1º en España *Andalucia*
 - Caract.: *Pagin.*, *GUI X11* (ver), *Plan. Multitar*. Apropiativa o Preentive, 4 *Tabla* básicas (ver). *POSIX Arch. Dispos. INode*.

Ver *Hist. Linux*

- **Android** :
 - El 70% de usr. Para *Movil* y pantallas tactil. Es *Linux Embbed*. Desarrollado por *Google* actualmente y basado en *Linux*
 - Arq. Softw.: a) *Kernel de Linux* (separa pila softw de pila hardw). b) Librerias y *Android-Runtime* (como *Maq. Virt.* de Proc. de *Java*). c) Armazón de Apl. (herram. de *Desarr.* de Apl). d) Apl.
 - *App Inventor*
 - *Plan.* 1er plano.
- **Mac OS** (orientado a su hard, ej. Mac OS Big Sur). Ver *AppleTalk (IPX/SPX)*, *Conect. Thunderbolt*
- **SO IBM** (International Business Machine): **z/OS** (para *Mainframe*). Otros *Hist. SO*: *OS/360* (1963 Mainframe, 1º *MultiHilo* y *Acces. SAM*) 1965-72, *MVS* (con *ISAM*, no confundir con *OpenVMS*). Ver *VSAM*, *Job Control Language (Cluster)*, *DB2 (SGBD)*
 - Destaca por PCs 1981, *SQL*, *QBE*, *Fortran*, *CICS* y *Supercomput.* (Deep Blue..), *Cuant.*, *Tarj.* perfor *BCD Texto* (Intro *MS*). Ver Acuerdo 82 (x86), *Hist. Win*, *Spooling*, *Mills (Estr. y Dis. Top-Down)*
- Otros:
 - *SO Servid.*: *Win Server 22* y *Ubuntu Server*, *Netware..*
 - *NOS*: *CISCO IOS*
 - *Movil*: *iOS* (originalmente para *iPhone*, luego en *iPod*, *iPad* y *Apple TV*). *Win10Mobile* (desastre, no soporte actual).
 - *Cloud OS (Nube)*: Ej: *ChromeOS* https://en.wikipedia.org/wiki/Cloud_operating_system



Chapter 18

+*16. Sistemas operativos: Gestión de procesos. (20OctEncarni)

18.1 Introducción. Conclusión

- Ej. Motivador Plan.: cola supermercado, familia (15min), mujer (bolsa, 5min) y jovén (bocadillo, 0.5 min). Satisfacción promedio: *FIFO* (2 cabreados 1 contento), *SJF* (1 medio cabreado, 2 contentos)
- Ej. Motivador Concurr.: *Semaforos* (*Telegrafos*) compartiendo un *Recurso*
- Hist. SO: MultiHilo Micro OS/360, Stallman *POSIX*, [Dijkstra65] *Filos./Trenes* y SO THEOS68 *Semaforo*, Coffman71 *InterBloq*.
- Conclusión: *Plan.* Proc. una de las 3 F. básicas de Micro Arq. *Kernel*.

18.2 Sistema Operativo y multitarea

- Arq. Sist. Informat.: Bibl. y API.
- Arq.s del SO: Micro Arq. Kernel las F. 3 básicas (*Plan.* Proc...)
- Multitar. y MultProgr..

18.3 Procesos y PCB

- **Proc.** eso (no confundir con *Hilo*): *Instanc.* de un *Progr.* que está siendo *Ejec.* por uno o varios *Hilos* (Fig. 18.16). Son la *Ud. min* del *Plan.* Proc. que cambiar de *Est..* Hay P. ligeros y pesados según *Recursos* consumidos (*Instr.* en *RAM* o *CPU*), pero todos son *Ud. min.* *Plan.* Proc. por SO NOTA: está en proceso, no es el “se dice ni se muestra” Wittg. de un *Progr.* o *Alg.*, puede haber varias *Instanc.* de un P.
 - Ej. *Daemon systemd* 1er Proc.
 - Otros: **Procesador** (o *Comput.*): el que ejecuta el Alg. *CPU*

- Otros: P.: *Alg.* que toma datos de entrada y devuelve salida o *Prod.*
- Otros: **Procesamiento** de Datos o *Signal*: el acto de modificación.

Es un *TAP Unif.*. Ver *Buffer*, *T. Ejec.*, *Eval.*, *Prod.*

- **PCB** o BCP (*Block Control Proc.* o *Descr.* de Proceso, no confundir con *Placa*): *Tabla* creada con el P. y donde el SO almacena toda la Info que necesita sobre este para *Plan.* Proc.: **PID** (*Id.* del Proc.), *Est.* en *CPU*, *Prior./Privelegios*, *Tipo Plan.* Proc. y *Gest. Mem.*. Es una de las 4 *Estr. Dat.* de los *SO* y se pone en una zona *Segur.* de *Mem.*, después de *Pila Llam.*.
- *Est.* de *Proc.* ver Fig. 18.16:
 - *Interr..*
 - *Cambio Ctx* /desalojo: ocurre cuando la CPU deja de ejecutar un P. para ejec. otro (ej. por *Interr.* ver). **Pierde t. en Carg.** nuevo *PCB*. Proc. del *Swap*. Ver *RT*
 - Listo a Ejec.: el *Alg. Plan.* Proc. elige entre los P. Listos y **Dipensador** lo mete en *CPU*. ver Fig. 18.16
 - Ejec. a Listo: Proc. demasiado t. en *CPU* y le toca a otro.
 - Ejec. a *Bloq.*: por una *Interr.* de Perif.
 - Tick y Bloq. a Listo: el *Buffer* del Perif. (audio o red) esta lleno puede pasarse a Listo el P. Si no, el SO cada tick (15 ms) comprueba *Buffer* y cuando está vuelve a calgar el proceso. Pg. 2, 15, 38
 - Inicio y Final: al diagr. podemos añadir dichos estados extremos clásicos.

18.4 Fork, Hilo (hebra) y multihilo

- *Fork* (no confundir con *Hilo*): *Concurr. RT*
- **Hilo** , Hebra o Thread (no confundir con *subProc. Fork*, ni *Threat=Amenaza (Vulner.)*): *Compo.* de un Proc. del que *Her.* su *PCB* (*Jerarq. padres-hijos*) y que es el conj. min. de *Instr. Manej.* por el *Plan.* Proc. para hacerles *Concurr.*. Un Proc. puede crear y supervisar varios H. alrevés creo que no. Por esta Her. tarda menos *Tmp* en crearse y finalizar que un P. (o subP. Fork). Tarda menos t. en comunicarse *Concurr.* con otra H. compartiendo *Mem.* mientras que los P. pasan por el SO.
- *MultiHilo*: poder ejecutar *Concurr.* varios H. en 1 sola CPU. Una de sus formas más famosas es el SMT (llamado Hyper-Threading en *Intel* ver *Superescalar*).
 - Usados en *Plan.* Proc. *Multiproc.*

Ver *Multiproc.*, *Java*, *TLP Pipeline*

18.5 Gestión de procesos. IPC

- **Gest. Proc.** (Managment): Funciones:

- *Plan.* Proc.
- *Cambio Ctx Est.:* crear (*Carg.*) /eliminar/ *DesBloq.* Proc.
- *IPC* y *Sincr., Concurr.*
- QUITAR?? *Sincr.* los Proc. adecuadamente: a) *Excl.* Mut.. b) *Esper.* Acotada (evita *Inanic.*) c) Progreso (si ningún Proc. en Sec. *Crit.* elegir a ejecutar entre los Proc. solicitantes, *Plan.* Proc.???)

18.6 Planificación de procesos. Algoritmos

- *Plan.* Proc. (Proc. **Scheduling**, no confundir con P. *RT*): una de las 3 F. básicas de Micro Arq. *Kernel*. Consiste en repartir *Concurr.* el T. *Ejec.* de *CPU* en ambiente de MultiProgr. (*Multitar.*, T. *Compart.*) entre los *Proc.* (o *Hilo*) que lo requieran. Conviene *Equil.* de *Carg.* para *QoS* adecuado y evitar la *Inanic..* El Kernel hace un “cronograma” usando la Tabla *PCB* (con T. Llegada, T. *Ejec.* y Prioidad):

- *Metr.:* ver abajo (*T. Resp.*)
- **Alg. Plan.** (o Disciplinas de *Cola*): ver bajo, distinguimos 2 No/Apropiat. Otros P. *Disco* (*SCAN, Brazo*).

Según Juanbe no necesaria desde los *Multiproc.* (**IMPORTANTE**) Ver *Proyec.* (*Diagr.*, *Topol.* Red), *Desarr.*, *Rut.*, *Crit.*, *TTL* timeout), *Opt.* *Consult.*

- *Metr.* de *Plan.:*
 - **T. Resp.** : *Metr.* t. desde que el proceso llega hasta que se obtiene resultado. Ver *RT*, *Plan.* *Disco*(*SCAN*), *QoS*, *Latencia*.
 - **T. Espera:** suma de t. que no está en *Ejec.* (suma de rosas claros)
 - T. Medios.: *Metr.* de *QoS*, cumple *Equil.* *Carg.* (Imparcialidad) y *Efic.* (Rendimiento).
- Alg. Plan. Apropiativos (o Preemptive): el *SO Interr.* el *Proc.. Multitar.* donde el *Cambio Ctx* lo hace el *Plan.* del *Kernel*. Lo llevan los *Kernel Linux* desde 2.4. Puede dar *Inanic.??*
 - **FIFO** o FCFS (First Input First Output o First Come First Servic.). Ver *Pagin.*, *Cola*, *Pila*
 - SJF (Shortest Job First): si hay varios a elegir, elegir al que tenga menos T. Ejec. Similar a Alg. Monótono (RMS *RT*)
 - HRRN (Higest Response Ratio Next): *Metr.* basada en los 2 anteriores el que lleve más T. Espera y con menos T. Ejec.
- Alg. Plan. No Apropiativo (o *Cooper.*): el Proc. se *AutoInterr.* (no el *SO*) y *Cambio Ctx* (ej. cuando este pasa a *Bloq.*). Es con perdida de t.
 - SRTF (Short Remaining Time First): en cada Tick o Quanto ver si hay alguno con menos T. Ejec. para terminar
 - RR (Round Robin o Rotatorio): en cada Quanto ($q = 2$) se cambia de P. De tipo Mejor Esfuerzo.

- *Colas MultiNivel*: de *Prior. Dinam.* (no como el Monotono RMS *RT*). Divide a los P. en colas según *Prior.* y cada cola tiene su propio Alg. Plan. Con **Retroalim.**: si un P. es muy usado aumenta su *Prior.* y cambia Cola. Ej. *Win.* .
- *Plan.* Proc. en Sist. *Multiproc.* (no confundir con *Gest. Mem.*):
 - a) *Equil.* de *Carg.* (los Proc. de cola van saliendo a la 1^a CPU disponible).
 - b) Asignación de CPU dedicado (opuesto al Eq. Carga, cada Proc. toma un n° de CPU según cantidad de Hilos y hasta que no terminan los H. no sueltan la CPU). **ASimetra**.
 - c) Por *Grup.os* (se buscan *Hilo* afines y se lanzan a la vez).
 - d) *Dinam.* (nº hilos de un Proc. cambia durante su ejecución) o *Prior. Dinam.* (ver *RT*)

18.7 Análisis de planificabilidad en tiempo real

- *Plan. RT: Prior. Estatica* (no *Dinam.*): Monotono RMS (Rate Monotonic Schedulling) (*Metr. Layland73*) con
Priot. Dinam.: EDF

18.8 Concurrencia

- *Concurr.* (no confundir con *Paral.* o *Secuenc.*): es *Part.ir* en *MultiProgr.* (con *Fork* o *Hilo*) + *Comunic.-Sincr.-IPC* entre Progr. (con Mem. *Compart.* como *Semaforo* o Pas. *Mens.* como *Signal*). Es un tipo de progr. *Modul.* que consiste en *Fact.* (o *DesCompo.*) un *Probl.* en un *DiGraf.* (*Rut. Crit.*) y Ejec. en *Ord. Parc.* sin afectar al *Prod. final*.
 - *Tma*: Todo Sist. *RT* es *Virt. Concurr.* (ver).
 - Ej: *Plan. Proc., Expr., SPN, Arbol de Op.*). Permite la comput. **Paral.** aunque no siempre se haga (como en T. *Compart.*). Ej: *Proyec. Cooper.* (ojo al caos *Aleat.* si mala *Comunic.-Sincr.*).
 - *Hist. SO*. aunque viene de la *Telegrafía* y ¿como varios ferrocarriles pueden usar eficientemente la misma vía (de ahí los semáforos y evitar *Colis.*)? fué *Dijkstra* el primero en informática.
 - Otros (ver abajo): *Probl. de C. (Cond. Carr.), L. C. (RT)*
 - IDEA: porque *Turing* es Secuenc. se olvida del poder de la *Concurr.. Lin.=SerieLizar.* Ver *Recon.* y *Atom.* Requiere *Comunic. Turing Complet. (Manej.)*.

Es *Unif..* Ver *NN, Emerg., Multiplex., MetaBiolog., GPU, Masiv., Hilo, POSIX, Multitar., Cond. Carr., Dijkstra, Bash* ampersand.

- *Calc. Pi π*: modelo mat. de L. *Concurr.* (y sobre todo el *Pas. Mens.*) mientras que Calc. *Lamb.* es para L. *Secuenc..* Es el Mod. del Paso *Mens..* La T. *Cat. Unif.* los distintos modelos de C. [Nielsen93]: Calc. Pi, Red **Petri** (*RT*), *Model.-Actor* (ver *Pas. Mens.*), *Comunic. Proc. Secuenc.* Ver *Turing, CODE2, Migr., Memoización (Efic.)*

- L. C.: son de *RT*
 - *LabVIEW* : Progr. *Visual*, L. ADA (DARPA). *Event*.
 - *ADA* 80 (basado en PASCAL, para unif. cod. de sist. *Embeb.* de *RT* de DARPA, *Concurr.*, por Ada Lovelace *Hist. L.* la 1^a progr. *Hist. L.*)
 - *C++ tcStd::thread, Event*.
- **RESUMEN** de los *Probl.* y Sol. de la *Concurr.:*
 1. La Sec. *Crit.* de un Alg. *Concurr.*, donde trabajan varios *Hilos* en *Paral.* como en una *Expr. SPN*, puede dar *Cond. Carr.* por las *Latencias* en propagación de *Signal*.
 2. Una sol. a las *Cond. Carr.* es la *Excl. Mut.* normalmente basada en *Esper.* Activa tanto para Softw. como Hardw. Distinguimos:
 - E.M. Hardw.: TSL
 - E.M. Softw.: Alg. Peterson que usa solo Mem. *Compart.*
 - E.M. SO: mediante *Dispos.* como *Semaforos* (un tipo de *IPC*) y *Monit.*
 3. Los Semáforos en Excl. Mutua y otras 3 *Condic.* Coffman pueden dar *InterBloq.* (\Rightarrow *Inanic.*) porque los Proc. se esperan unos a otros .
 4. Otras causas de *Inanic.* son Err. en *Plan.* Proc. y *Ataq.* Dos.

18.9 Problemas de la concurrencia: Sección crítica y condiciones de carrera

- Sec. *Crit.* : Sec. de Código (o conj. de *Instr.*) de varios *Proc. Concurr.* donde se *Acces.* a un *Recurso Compart.* (como *Estr. Dat.* o *Perif.* o *Mem.* en *Transac.*) y que si no se cuida pueda dar *Cond. Carr.*. Lo normal es que solo puede acceder en cada t. un solo Proc (sin *Interf.*). Ej: *Transac.* [0,1,1,0], Fig. 18.16. Ver *SGBD*, *Rut. Crit., Prior. C. (RT)*
- *Cond. Carr.* (Race Cond.): problema surgido de la Sec. *Crit.* donde la O. de un *Sist.* puede ser durante unos instantes diferente a la esperada por la falta de *Sincr.*. Debido a *Latencia* (retardo en propagación *Luz*) y la Ejec. *Ord. Parc.* de sus *Compo.* (en *Concurr.*). La solución es la *Excl. Mut..* Distinguimos:
 - C. C. por Softw: a) Si un Progr. tiene varios *Hilos* (Cod. Ejec. *Concurr.*) y estos terminan t. distintos a los esperados. b) Si varios Proc. R/W un *Recurso Compart.*
 - C. C. por Hardw.: *Secuenc..*

Ver Alg. Dekker, BiEst.

18.10 Soluciones: Exclusión Mutua. Espera Activa.

- *Excl. Mut.* (**mutex**, no confundir con *Excl. Prob.*): en Prog. *Concurr.* es un requerimiento en el que se evita que dos Proc. entren en Sec. *Crit.* al mismo *Tmp* y psviene de *Cond. Carr.*. Ver abajo Soluciones Softw., SO y Hardw. Ver *Transac.*, *SGBD*.
 - *Dispos.* usados para *Sincr.:* *Semaforos*, *Monit.* y *Pas. Mens.*

- *Sincr. Condic.*: un Proc. puede Op. \iff otros Proc. están en un *Est.* (o han hecho otra Op.)

Ver *Arch. Dispos.*, *Compart.*, *Op. Log.* *XOR*, Terc. E. (*Axiom.*).

- *Esper.* No/Activa o *I/O A/Sincr.*: uso de *Buffer*.

18.11 Comunicación: memoria compartida (semáforos, monitores)

- Mem. *Compart.*: una de las 2 formas de *Comunic.* en *Concurr.* siendo el *Medio* que tienen los *Proc.* para *Comunic.*. Es *Unif.*. Destacamos: *Mutex (Excl.)*, *Semaforo y Monit.*
- *Semaforo* (un tipo de Cerrojo o Lock): es una Sol. a la *Excl.* mutua. *Var.* o *Tip. Dat. Abstr.* que lleva la cuenta del nº de Recursos disponibles para *Concurr.* en *IPC* y *TSL* (y Alg. Dekker como cerrojos). Se *Manej.* solo con 2 F: a) $signal(S) = S++$ (crece S.) y b) $wait(S) = S--$ Si $S > 0$ (pausa Proc. hasta que $S > 0$ y entonces decrece S.). Similar a Bandera-Contador-Centinela en *Iter.???*. Posible *Inanic.* y *Cond. Carr.* (si se cuela Proc. antes de tiempo). Inventado por *Dijkstra*62 para el su *THEOS*

Imagina 10 habitaciones para estudiar (Recurso) y muchos estudiantes esperan a entrar (*Hilos*) \Rightarrow valor inicial del *Semaforo* $S = 10$. Si $S > 0$ entra estudiante y $S--$. Si $S = 0$ estudiante espera cola *FIFO*

- *Monit.or*: *Cl.* o *Estr. Dat.* *Mutex (Excl.)*, donde los distintos *Hilos* solo pueden Ejec. un *Metod.* al mismo tiempo. Evita inconveniente de *Semaforo* de que las *Llam.* a F. están *Distrib.* por Código, siendo poco *Eleg.* su *Manten..* Por *Hoare Hist. SO* (*Ord. QuickSort*). Ver *RT*.

18.12 Comunicación: paso de mensajes

- *Pas. Mens.* : técnica (casi *Parad. Progr.*) donde los *Proc.* se *Comunic.* que hacer. Su modelo es el *Calc. Pi* y el *Model. Actor.* Empleada en *Concurr.*, *Event.* y *POO* (*Filos. Kay Polimorf.*), *Proc.* o Actores dentro del mismo Progr., SO o incluso Comput. (Obj. Distrib. *Progr. Compo.* (CORBA)).
- P. M. *Sincr.*: emisor envia y no continua hasta recibir *AcRe.* ASincr. lo contrario.
- Ej: *Pipe* (abajo). Ej: Creo: *Signal SIGNUSR1 (Interr.)+Manej.* con *Pipe* como mi Proyecto

Ver *Tip. Dat. Abstr.*

- *Pipe* (Tubería, no confundir con *Pipeline*): es una forma de *Pas. Mens.* y *Op. Comando*. Es un *Fich.* para *IPC* y *Compo. Comandos*. A veces llamados *Cola FIFO* pues es un *Descr. Arch. (INode)*,
- P. en mi Proyecto *RT*: para *Concurr.* es una *Var. Compart.* pero creo *Sincroniza Actual.* (evita *Cond. Carr.* con *Mutex (Excl.)*, no *Actualizazo* hasta termina de escribir)

18.13 Soluciones basadas en Exclusión Mutua

- Soluciones Softw.: Alg. de EM
 - **Alg. Dekker** (o de los 2 Proc. o del *Semaforo* del Baño/Trenes): usa Mem. *Compart.* (MC). Cuando un Proc. entra en Sec. Crit. (baño) pone el *Semaforo* de MC a 1 y al salir lo baja a 0. **Defectos:** Solo 2 Proc., *Esper.* Activa y posible *Cond. Carr.* haciendo que se cuele otro Proc. cuando no debe. [Libro Dijkstra], antes en trenes compartiendo segmento *Hist. SO*
 - Alg. Peterson (o de los N Proc.): Alg. para *Excl. Mut.* basado solo en MC que mejora Dekker para N Proc. MC la *Tupla* $T = (\text{bandera}[N], \text{turno}[N - 1])$. El Proc. i antes de su Sec. Crit. llama a *Entrar*(T, i), esto bloquea al Proc. (*Esper.* Activa *while*($F(T, i)$) hasta que se sale del *while* y se ejecuta Sec. Crit. Cuando finaliza llama *Salir*(T, i) y el resto pueden acceder a Sec. Crit. **Defecto:** *Esper.* Activa.
 - Otras: Alg. Panaderia Lamport.
- Métodos Hardw. para lograr EM
 - *Masc. Interr.* (Deshabilitar, Inhabilitar o Disable): forma más simple de conseguir *Excl. Mut.* El Proc. antes de entrar en Sec. Crit. De esta forma ningún proceso puede Interr.. Una vez termina las deshinibe de nuevo. **Defecto** puede dar fin al sistema.
 - *Instr. TSL (Test & Set Lock)*: basado en *Esper.* Activa (**Defecto**), ningun otro *Proc.* puede hacer un TSL en una celda de *MP (Var.)* hasta que otro termina su TSL. Permite crear *Semaforos* (=cerrojos) como Dekker pero sobre MP Hardw.. En x86.
 - Alg. Peterson: en Hardw. que bloq. el *Bus* soluciona la *Excl. Mut.* de SMP *Multiproc..* **Defecto:** Espera Activa
- Sol. SO de EM: mejor para *Sincr.* complejas donde las sol. sofware y hardware basadas en Mem. *Compart.* son incontrolables:
 - Dormir y Despertar: para solucionar *Esper.* Activa de *Test-Set* y Peterson el SO suspende-bloquea el Proc y lo despierta hasta que otro lo hace con una *Signal.* Sleep, wake, wait (*POSIX?*) para no consumir Recursos.
 - *Semaforo, Monit.* y Paso Mens..

18.14 Problemas de los semáforos: Interbloqueo e inanición

- InterBloq. (Deadlock o Bloq. Mut., de *Transm.* o *Proc.* no confundir con *Block Mem.*): ningún Proc. puede acceder al *Recurso* porque espera que otro lo haga circularmente, Fig. 18.16. Interbloq \Rightarrow *Inanic.* pero el *Recipr.* no es cierto (la I. puede ser por otras causas ver). Solución romper la simetría.
 - **Filos. Chinos:** ej. de InterBloq. de [Dijkstra65] *Hist. SO* si cogen los palillos a la vez se Interbloq. y mueren por Inanic. Fig. 18.16.

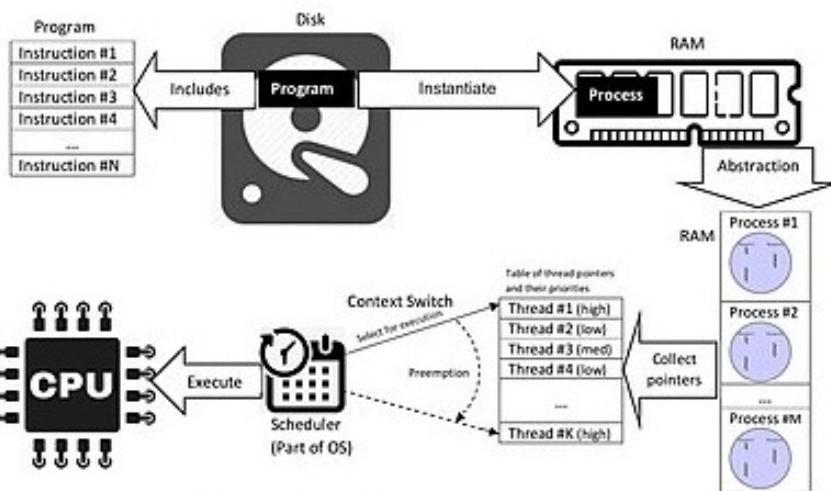
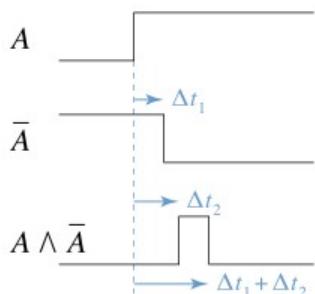
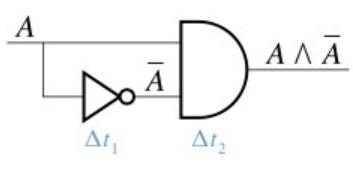
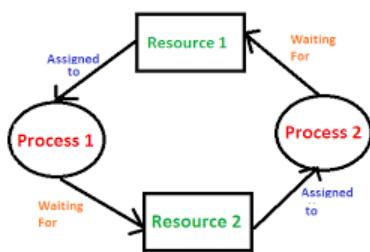
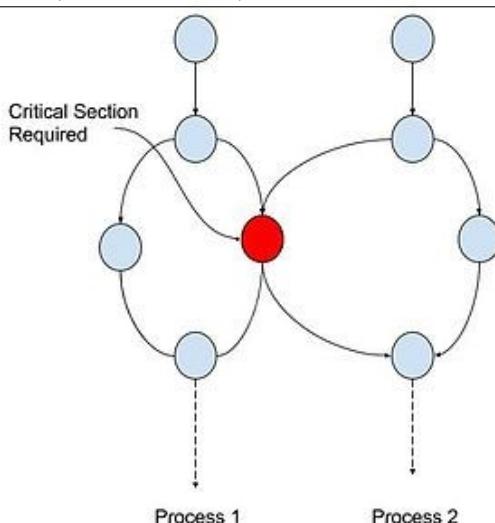
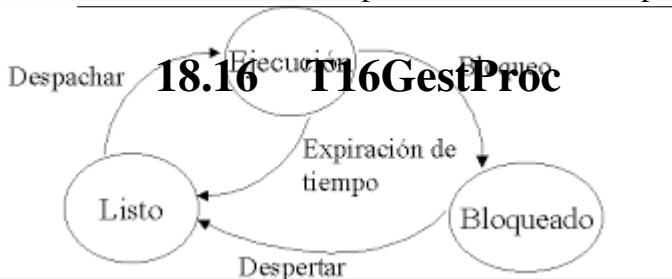
- **Inanic.** (Starvation, no confundir con InterBloq.): cuando a un Proc. se le deniega continuamente el acceso al Recurso para que pueda seguir trabajando. Suele ser debido al InteBloq. pero también a una mala Plan. Proc. (ver SCAN Brazo) o a un Ataq. DoS. Sol1: Plan. Proc. no apropiativa. Sol2: **Esper. Acotada:** limitando el nº veces que otros Proc. ejecutan su Sec. Crit.

Ver Est. del PCB, Recep. B. (RT), Interr., Congest.. Esper. Activa.

- **4 Condic. [Coffman]71 Hist. SO** del InterBloq. \iff :
 - 1) Excl. Mut. (*exists Recurso (R.) Compartido pero que solo puede usarlo un Proc. (P.).*)
 - 2) Esper. y Retener (mientras uno usa R. los otros esperan).
 - 3) Aprop. o No Expropiación (el R. es solo liberado cuando termina el P. y si el P. se queda en un bucle Iter. no liberará nunca el R.).
 - 4) Espera Circ.ular (unos esperan a otros)
- Soluciones: a) Evitar que se produzca: Preveer (evitar 1 de las 4 cond.) y evitar (solicitar info a Proc. para saber si es seguro darle el Recurso). b) Reparar una vez ocurrido: Detección (periódica por si ha ocurrido), Romper simetría (dejar que uno actúe) o Eliminar (todos los procesos y empezar)

18.15 Gestión procesos en SO populares

- *Win*: pensado para responder: a) a 1 usr con mucha interactividad b) a un servidor. Tiene un Programador Preventivo (InteBloq.???) y Plan. Proc. basada en Colas MultiNivel (dentro de cada nivel Round Robin con Prior. Dinam. como en RT). La Ud. de Plan. es el Hilo no el Proc.
- *Linux*: desde el Kernel 2.6 tres políticas: 1) Para subproc. en RT FIFO. 2) subproc. en RT Round Robin, 3) Hilos de RT usa Round Robin
- *Android*: 2 tipos de Proc. a) Nativos y b) Ejecutados en JVM. Hace Plan. Proc. Apropiativa con Round Robin con Prior. (orden: activos, visibles, Servic., 2º plano y varios). Si es necesario Mata, pausa o libera Recursos para Apl. más Prior. Ej: Cierra un Proc. que no se ve en Pantalla y deja el que se ve.
- *Mac OS X*: planif. colas multinivel + Prior..



Program vs. Process vs. Thread
Scheduling, Preemption, Context Switching

Chapter 19

*17. Sistemas operativos: Gestión de memoria.

19.1 Introducción. Conclusión

- Ej. Motiv.: tenemos un espacio límitado (el armario, trastero o almacén) ¿como ponemos las cosas? (Sol: las que usamos más amenudo más a la mano) ¿Y si tenemos huecos ya establecidos? (Sol. las más grandes primero para evitar *Fragm.*)
- Ej. Motiv.: si queremos aprovechar los Recursos de CPU debemos usar *MultiProgr.* Aunque Maq. *Turing*: tiene mem. infinita las nuestras no. y la que demandan los *Proc.* es impredecible.
- Ej. Motiv.: Como ir almacenando la reserva de energía en seres vivos (*Biolog.*).
- *Hist. SO*: ver Atlas *Manchester 1º Pagin.* e *Hist. Linux*
- *Actual.*: ver *Swap*, *x86 Pagin.*
- Conclusión: *Mem. Virt.* una de las 3 F. básicas de Micro Arq. *Kernel*.
- Futuro: no *SO*

19.2 Fundamentos: sistema operativo (Kernel), jerarquía de memoria y multiprogramación

- *Arq. Sist. Informat.*: Aplic. *SO Kernel Hardw.*
- *Micro Arq. Kernel*
- No *MultiProgr. (Multitar.)*: desperd. *Recurso*, aquí *Mutiprogr.*

19.3 Hardware. Controlador de memoria. MMU

- *Mem. y Jerarq. Mem.*
- **MC** (*Memory Controller*): Genera las *Signal* para comunicarse con *MP*. Hasta 2008 un *Chip* externo o en P. Norte. (ver **Norte=MC Hub**). *Actual. Integr.* en el *Micro* llamado iMC. Ej: el *MMU* o el *MC* de DDR SDRAM (ver). Ver *Bus Hypertransport, VRAM e iGPU*.
- **P/MMU** (*Pagin./ Mem. Management Unit*): Gest. de *Mem. Virt.* cuya F. principal es *Traduc. Direc. Virt. Log. (Pagin.)* a dirc. *Fisic.* (Marcos Pg.) usando la *Tabla Pagin.* (ver). Es un Hardw. *Integr.* en *CPU* (ver *x86 Pagin.*). Otras funciones son:
 - **IMPORTANTE:** Cache 2 significados???: Control de C. [Wiki] y **TLB** (C. de *Pagin.*)
 - Protec. Acces. Mem. (ni un malware ni otros Proc. pueden usar memoria de otros Proc. sin su *Permiso*)
 - *I/O MMU*: ver *Video RAM*.
 - Mem. *Compart.*: ej. para *IPC*
 - Quitar/Meter *Progr.* en *MP*

Ver *Swap. x86 (Segment.)*, Ud. *Control (Registr. MDR y MAR???)*.

19.4 Gestión sin virtualizar (asignación): tabla y métodos

- **Gest. Mem.** (*Memory Management*): de *MP* sin necesariamente *Mem. Virt.*. Funciones:
 - Asign. (ver abajo)
 - Control del *Esp. Libre*: Recol. *Basura* y 2x3 del T19 *Sist. Arch. Cuota*
 - *Mem. Virt.* (abajo)
 - Otras Lo realiza: a) Softw. el *Kernel* (en la *Asign.*). b) Hardw. el *MMU*.
- Ver *Carg. Ensambl.. CPU*.
- **Tabla de Asign.:** RESUMIR O QUITAR:
 - Tabla similar al *PCB*, *Descr.* las particiones (límites) con base, tamaño, *Est.* para controlar partes *Libres* y *Asign.*. Pueden ser muy grandes. Contiene también bits del historial: ej. *R* (zona referenciada) y *M* (zona modificada). Los *Registr.* Lím. (como en estática) *Almac.* en hardw. lim. inf. y sup. ???. Formas:
 - *Map.* o *Masc.* bits (como *Trim*): para zonas (0: si libre y 1: si ocupada)
 - *Lista Enl.*: Fig. 19.9. *Almac.* en mem. de zonas asignados y libres de mem.
- **Asign. Mem.** (*Allocation*): a los *Proc.* que la solicitan permitiendo un Gr. de *Multitar.* F. del *Gest. Mem.* basada en *Estr. Dat.* y mediada por el *Kernel*. Puede ocurrir *Fragm. Externa* (suma da pero no cabe). Estrategias
 - **MEMOTÉCNICO** (Piscina, colegas, losas resbala (slab), pila):

- A. Estat. o *Block Fijos* o Pool: se parte en bloques de tam. fijo (como *Pagin..*). Usa una *Lista libre* o Pool (piscina) que va rellenando. Útil en sist. *Embeb.* con pequeños Proc.. Por su simplicidad también en *Videojuegos* que A. y DeA. a gran Vel.. Da *Fragm..*
- A. Colegas (Buddy o Asociados): se parte Mem. en Pools de distinto tamaño (pot. de $2 \cdot 2^n$) y **se va llenando en Arbol.** Con *Lista Enl.*
- A. Losas (Slab): se parte en losas de tam. distinto (como *Segment.*) y mete los Proc. según convenga.
- A. *Dinam.* o *Pilas* (Stack) (ver): usa *Punt.s* como *malloc*. Típico de *Linux* y *Win*

Ver *Sist. Arch.*, A. *Esp. Libre* (*Sist. Arch.*), *CIDR, ICANN*

- RESUMEN de lo antiguo: *Asign.* Contigua de Mem.: obsoleto, a cada *Proc.* se le asigna una zona o mapa de mem. Ver *Carg..*
 - *Asign.* Fija o Estát.: en el momento de creación del SO se divide la memoria en particiones de tamaño fijo. Su tamaño es func. de: los Proc. más frecuentes en el SO, Gr. *Multitar.* y capacidad de memoria. Ej: *Pagin.??* Ventajas: ideal si procesos con tamaño conocido y en Proc. *Lotes*. Inconv.: si demanda cambia *Dinam.* puede dar *Fragm.* Ext. e Int.
 - *Asign. Dinam.:* se van creando particiones del tamaño del programa. Cuando salen los procesos se crean zonas del tamaño de los procesos entrantes. Solo se da *Fragm.* externa. pues ningún progr. ocupa más espacio del necesario. Usa *Tabla Mem.* Ej: *Segment.??*
 - *Asign. No Contigua:* gracias a la *Mem. Virt.* y la traducción del *MMU* no tienen que coincidir *Pagin.* y Marcos *Pagin.* El programa, los datos y la pila pueden exceder la **Mem. Física** (normalmente MP) disponible usando direcciones mayores y MS. Aumenta el Gr. *Multitar.*
 - *Asign. de los Colegas* o *Asociados* o *AGrup.ados* (Buddy Mem. Allocation): divid. la mem. en mitades para intentar ofrecer el mejor ajuste. Agrupa la memoria en fragmentos similares y cuando *Busq.* un bloque analiza los similares en tamaño.

19.4.1 Asignación (gestión sin virtualizar): Fragmentación

- *Fragm.* : no uso bien **Ajustado** y por tanto desperdicio de Mem. o *Esp.* (MP o MS). Es consecuencia de malas Políticas de *Asign.* de Mem como *Sist. Arch. FAT* (ver). Sol.: DesFragm.
 - F. Extern: $= \sum \text{huecos} > \text{TamProc, PeroNoEntra, EnAsignBlockVar(Segm.)}$. En mem. *Part.* en *Block* variables (como en *Segment.*), al ir desalojando *Proc.*, la suma de huecos da para alojar (*Asign. Mem.*) al *Proc.* pero no puede meterlo en un hueco entero. Fig. 19.9. Sol: compactación.¹
 - F. Intern.: $= \sum \text{huecos}/N = \text{HuecoPromedio, EnAsignBlockFijos(Pagin.)}$. En mem. Part. en *Block* fijos (como *Pagin.*), es la *Metr.* de desperdicio de mem. por asignar todo el bloque a procesos más pequeños que el bloque total. Sol: *Asign. Dinam.* de *Part.* Mejor Aj. (Best Fit Block Search)

Ver *Asign. Estat.* y A. *Sist. Arch.*

¹[Geeks] Tanto si aplicas una estrategia de asignación de memoria de 1er como mejor ajuste, causará F.E.

19.5 Memoria virtual

- **Mem. Virt.** (MV): una de las 3 F. básicas de Micro Arq. Kernel. Permite *Exten. Esp. Fisic.* o *Rango Direc.* que pueden *Manej.* los programas (*Virt.* mayor *Masiv.* que la *MP.* *Pagin.* y *Segment.*): las dos forma básicas de gestionarla gracias al *MMU* (o al *IOMMU*) y *Swap*. Ver Fig. 19.9.
 - **Swap**: zona de intercambio (antes *Part.*, hoy *Fich.* del SO *Sist. Arch.*) para mover programas entre *MP* y *MS*. Fundamenta la *Mem. Virt.* (MV). Mem. del *Cambio Ctx.*
 - Ventaja: aumenta Gr. *Multitar.* (mejorando el rendimiento del sistema) y ejecutar progr. mayores que la MP. Desventaja: hay que *Gest. Mem.* (ir quitando y ubicando trozos de memoria in/útiles).

Es *Unif.* por *Exten. Recurso*

19.6 Memoria virtual: paginación

- **Pagin.** acción: Mod. *Direc. Offset* (Fig. 19.9) que parte la memoria en trozos fijos denominados pg. **Usada en Win y Linux. Supercomput.** Ferranti's Atlas de Manchester 1960 y el *THE* de Dijkstra los 1ºs con Pagin. *Hist. SO*
 - **Marco Pg.**: zona de MP (*Fisic.*) que no tiene porque coincidir con la pg. o zona de MV. Ver *Asign.* No Continua y Fig. 19.9 .
 - **Nº Marcos Pg.=MP/TamPg.** Sec. o Cadena **Ref. Pg.** ej: 2,4,5,... Si hay de sobra el nº de *Fall. Pg.* es = nº de pg. distintas.
 - **Fall. Pagin.** (no confundir con Fall. Segment. (*Null*)): ocurre frecuentemente, cuando un *Proc.* quiere acceder una pg. y su marco correspondiente no está en MP. **Hiperpaginación** (thrashing) usar una cantidad de recursos cada vez mayor para cada vez hacer menos trabajo. Ocurre cuando un proc. tiene pocas Marcos Pg. y se dan muchos Fall. (entonces el proc. pasa más tiempo trayendo pg. que ejecutando instrucciones y bloquea otros).

Ver *MMU*.

- Políticas Reemplazo (o Alg. Sustituc.) en *Pagin.*: en MP de Pg. Intentan minimizar los *Fall. Pg.* Los alg. usan una **Tabla de Víctima a Reemplazar** (Fig. 19.9). En paréntesis ponemos un comentario.
 - **FIFO**: sustuir pg. que más t. lleva en mem. (puede quitar pg. importantes). (Disciplina *Cola*)
 - **LRU** (Least Recent Used): sustuir pg. no referenciada desde hace más t. (excelente pero difícil de *Impl.* exactamente).
 - **Seg. Oport.** (Reloj, con bit R de Referenciado o usado): (mejora FIFO, es realista)
 - **Aging**: sustituye la que lleve más t. sin usarse (se aproxima bien a LRU)
- Mejoras en la *Pagin.*:

- **TLB** (Translation Lookaside Buffer, *Buffer de Traduc.* Avanzada): una *Cache* ad-min. por la *MMU* que contiene partes de la *Tabla de Pagin.* que relac. direcc. físicas y log.
 - T. *Jerarq.* o *MultiNivel*: evita *Almac.* en MP enormes tablas, solo las necesarias. Ej: *x86-64*
 - Otras: *Cache* de entradas más recientes
- *Pagin.* en *x86-64*: la *Segment.* es *Legacy* y lleva *Pagin.* con Mod. Mem. Plana. Cuando se usa con 4kB de Pg. el *Arbol Tabla* es de 3 Nivel.

Sea una *Mem. Virt.* basada en *Pagin.* con 5 **Marcos Pg.** (inicialmente vacíos). Un *Proc.* genera la siguiente secuencia de referencias a Pg.: 25135, 04108, 79571, 025982. ¿Cuántos Fall. Pg. da LRU? (Sol. 15) ¿Y FIFO? (Sol. 14). ver Fig. 19.9

19.7 Memoria virtual: segmentación

- *Segment.* (no confundir con S. PDUsd.): se parte la Mem. en trozos variables *Dinam.* denominados segm. Es *Legacy* en *x86*
 - *Tabla Mem.*: hay 2, de segmentos y de segm. libres
- Ver Segmentation Fault (=Violación de Acces.=Vulner. Overflow Null).
- Polit. Ubicación *Segment.*: intentan minimizar los huecos o *Fragm.* interna.
 - Primero Ajustarse: elige el 1º que quepa en la lista conforme aparecen los H (Huecos) y lo tacha.
 - Mejor Aj.: busca en la lista de H y elige el mejor y lo tacha.
 - Peor Aj.: en el que más H interno o F produzca para luego recuperar por DesFragm.
 - Siguiente Aj.: recorre la lista H circularmente. El 1º encaja como en Prim. Aj. El 2º busca el siguiente primero que encaje.

Si en la MP tenemos los huecos (o particiones libres) en KB: 1000, 400, 1800, 700, 900, 1200 y 1500. Y 3 procesos de tamaño 1200, 1000 y 900. Para los alg. de *Segment.* Primer, Mejor, Peor y Siguiente Aj., indicar huecos asignados (H) y *Fragm.* (F).

Proc. Segmento	Prim Aj. (H/F)	Mejor Aj.	Peor Aj.	Sig. Aj.
P1 1200	1800/600	1200/0	1800/600	1800/600
P2 1000	1000/0	1000/0	1500/500	1200/200
P3 900	900/0	900/0	1200/300	1500/600

19.8 Paginación vs segmentación

	<i>Pagin.</i>	<i>Segment.</i>
Tamaño <i>Direc. Log. o Virt.</i> Ventajas/I	pg. fijo nº pg. + desplaz. No <i>Fragm.</i> ext y algo de int., facilita reubic. y protec.	segm. varia nº segm. + despl. No Frag. int. y a veces ext. (por tam. max. segm.). V: facilita compartir mem. entre procesos ej. <i>Bibl.</i> gráfica (en pg. más dif.) I: el programador debe dividir su programa en segmentos
Mejoria	<i>Tabla Mem. multiNivel,</i> TLB	
Otras:	Creo más para MP??	más para MS

- *Segment. Pagin.ada* (o Pg. Segm.): mezcla lo mejor de ambos. **Cada Segment. se divide en Pg. Direc.. Log.=nº seg+nº pg en segm.+despl. en la pg.** V: \nexists *Fragm.* ext y la int. controla con tam. pg. Soporte a compart., protec. y *Estr. Dat.* cambiantes. Ver ISAM

2	5	1	3	5	0	4	1	0	8	7	9	5	7	1	0	2	5	9	8
X	2	5	1	3	5	0	4	1	0	8	7	9	5	7	1	0	2	5	9
X	X	2	1	3	5	0	4	1	0	8	8	9	5	7	1	0	2	5	
X	X	X	2	1	3	3	5	4	1	0	0	8	9	5	7	1	0	0	
X	X	X	X	2	1	3	3	5	4	1	0	0	8	9	5	7	1	0	0
F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F

FIFO

2	5	1	3	3	0	4	4	4	8	7	9	5	5	1	0	2	2	8	8
X	2	5	1	1	3	0	0	0	4	8	7	9	5	1	1	1	0	0	
X	X	2	5	5	1	3	3	3	0	4	8	7	9	5	5	5	1	1	
X	X	X	2	2	5	1	1	1	3	0	4	8	7	9	5	5	5	1	
X	X	X	X	2	5	5	5	1	3	0	4	4	8	7	9	9	9	5	5
F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F

Process 07

needs 50KB
memory space

Fragment

40 KB

Assigned Space

Fragment

Used Space

Fragment

Used Space

Fragment

10 KB

Assigned Space

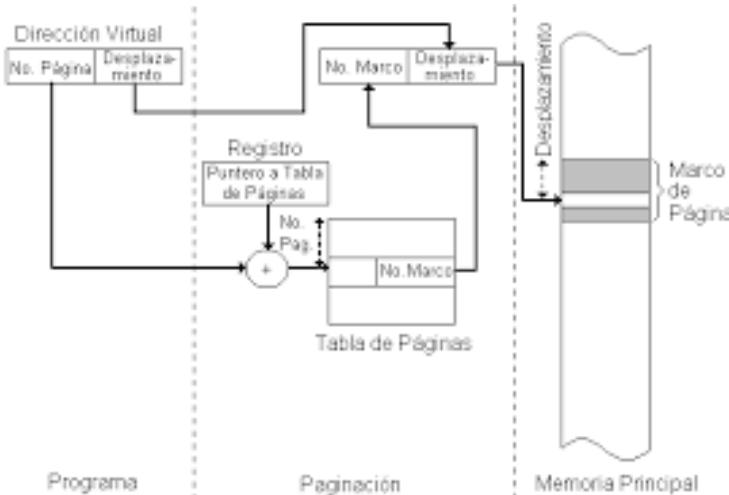
Assigned Space

Fragment

5 KB

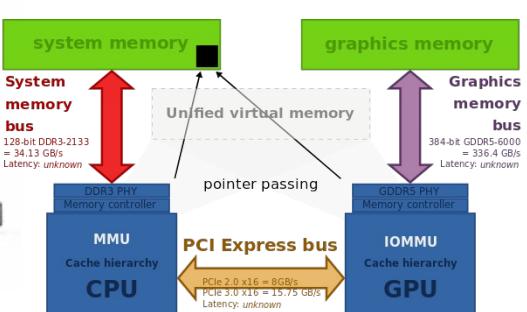
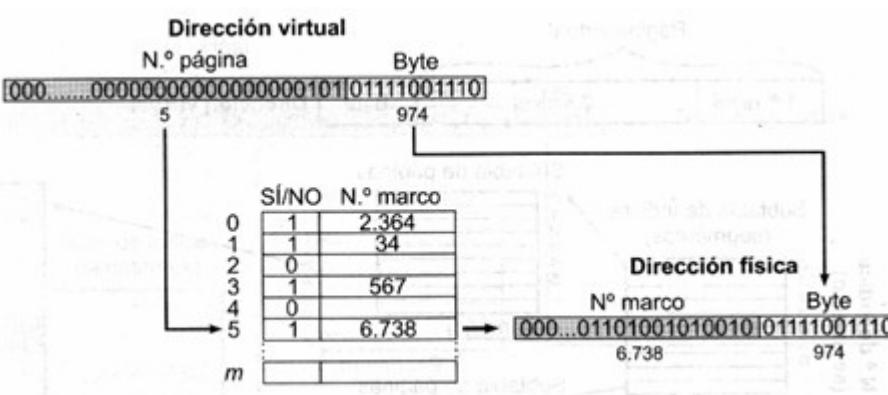
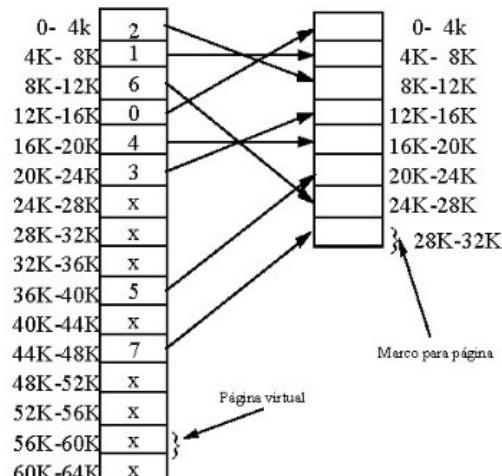
External Fragmentation

Internal Fragmentation



Espacio de direcciones virtuales

Direcciones en la memoria física



Chapter 20

*+18. Sistemas operativos: Gestión de entradas/salidas. (18MarJoAnt)

20.1 Introducción. Conclusión

- Ej: ¿Se puede seguir usando CPU mientras ocurre operación de I/O en MS? Sol. si con *DMA*
- Hist.: Super *I/O IBM*, *Tanenbaum Arq.* I/O, SPOOL System de *IBM* en 1962 1º con *Spooling Tarj.* Perfor. **Plug and Play** 95 Win
- Futuro: *IoT Dat. Masiv.*

20.2 Sistema Operativo

- *SO*: controla *Recursos*, intermediario Usr-Hardw, aisla de particular. de *Dispos.* (*Transpar.*).
- *Arq. SO*: Fig. *API*

20.3 Hardward de E/S: periféricos

- *Perif.* o *Dispos.*: de *I/O*.
- Perif. de Uso a) *Excl.*, b) *Compart.* y c) *Virt..*
- Perif. *Estr.* (*Est., Control y Dat.*).

20.4 Controlador vs driver

- *Controlador de Perif.* o (C. *I/O* o *Controller*, no confundir con *Driver*).: *Chip Impl.* en a) *Dispos.* b) *Tarj.* c) *Placa madre*. Ej. Super *I/O SouthBridge* de *IBM* Hist. *SO*

- *Driver* o *Controlador de Dispos.* (no confundir con C. de *Perif.* ni *API??*): conj. de F. *Estand.* del SO y *Llam.* por el Progr. *Usr* para *Controlar (Comunic.)* un *Dispos..* Fig. 20.13. Es un *Traduc.* o *Interf.* o *API* y está *Impl.* para un SO concreto. Posee un *Buffer* por si se le acumulan Comandos.
 - Ej: F. verificar *Est.* del Dispos.
 - Ej: *devmgmt.msc*, *Admin. Dispos.: Instal. D. Tarj..*
 - *Firmw.:* corre en el D. y el *Driver Comunic.* el SO con el Dispos. Ej.: *Tarj. Red (Softw. Red)*.
 - D. Hardw.: para *Perif.* o *Robot* (ver)
 - *ALSA* (Advanced Advanced Linux Sound Architecture): *API* para *Driver de Tarj.* Sonido que está en *Kernel Linux* (OSS ya no se recomienda en *Ubuntu*). usado por *Bibl. PortAudio*

Ver *Middlew.*, *Petic.* IRP., *Comando Trim (SSD)*, *Conect. BD*

20.5 Gestión de E/S

- Gestión de *I/O* : la 4º F. básica de la MicroArq. *Kernel* (no incluida). La Arq. Softw. de I/O de *Tanenbaum (Hist. SO, MINIX Hist. Linux B., Flynn)* distingue 5 niveles y F. (dcha) de la Fig. 20.13 (como mi Proyect.):
 1. *Softw. Usr* (\notin SO, **Progr. C**): *Bibl.* de IO *Enl.* por Progr. *Usr*. Su F. principal es hacer *Llam.* a IO.
 2. *Softw. Independ. Dispos. (API PortAudio ver ALSA)*: F. nombre, uso de *Buffer+asigna Driver*, tamaño *Bloq.*, info de *Err.*, libera dispos. Ej: *Unix usa Fich. Especiales* \forall Dispos.
 3. *Driver o Manej. de Dispos. (ALSA ver Bibl. PortAudio)*: implementado para un SO concreto (ver).
 4. *Manej. de Interr.:* depende del Hardw. *Proc. \rightarrow Desalojo – Kernel...* (ver abajo)
 5. *Hardw.:* Control. *I/O* (ver, C. *DMA*, C. *Tarj...*).

Es *Unif..* Ver *Canal IO Perif.*, *MIO*, *Dispos.*, *Puerto*, *Canal. MMIO (TPerif.)*, *IOMMU*, *I/O ASincr. (Interr. o Esper.)*

- Gest. *I/O*: . Las F. son:
 - *Acces. Transpar.:* el SO hace que los *Perif.* interac. con los *Proc.* sin que se preocupen por detalles.
 - Limitar Acces.: de Proc. a partes de Perif., ej. a *Discos*
 - *Manej. Interr.:*
 - *Plan. Acces.:*
 - Evitar *Cuello Botella: Buffer..*

20.6 Mecanismos básicos de gestión de E/S: spooling y chaching

- *Buffer*
- *Spooling* (Simultaneous *Perif.* Operations Online, no confundir con Polling o Sondeo de *Esper.* Activa): es *Filos. Petic.* (sist. a dist. *Vel.=Cola+Buffering*), el *SO* pone trabajos o *Petic.* (del mismo Sist. o de otros PCs) en un *Buffer* y el *Perif.* accede a el cuando esté listo.
 - S. en MultiProgr (*Multitar.*): permite a *CPU* y b) **Virt.ualizar o Compart. Dispos. de uso Excl. Ej: Servid. de Impr.** (ver *Daemon*)
 - Lectores de *Tarj.* perforadas, el 1er SO con Spooling fué: el SPOOL System de *IBM* en 1962 (*Hist. SO*).
- *Cache*: conect. dispos. a dist. *Vel.* y Dat. Frecuent.

20.7 Técnicas de gestión de E/S: PIO (programa), IIO (interrupción) y DMA

- **PIO: P/MMIO**
- **IIO (I/O por Interr.):** mejora *Efic.* de PIO (*Proc. → Desalojo – Kernel...* ver I. *Cambio Ctx*) pues hasta que se recibe *Signal* del *Perif.* de “estoy listo” la CPU hace otras cosas, es **I/O ASincr. (no Esper. Activa)**.
- **DMA** (Direct Mem. Acces., no confundir con Acces. *Aleat.*): mejora PIO e *Interr.* que no son *Efic.* (por *Vel.* limitada de CPU y ocuparla). El *Sist.* permite a *Perif.* Acces. a *Mem.* sin pasar por *CPU* hasta que esta recibe *Interr.* del *Controlador de DMA* de “**listo**”. La CPU solo interviene para autorizar la *Petic.* de Transferencias. Es para Dat. *Masiv.* y para copia de *Bloq.* a *Bloq.* de un *Dispos.* a otro.
 - Implica *Cuello Botella* por ocupar el *Bus* y CPU mientras lo necesita, además paraliza el *Proc.* (no *CPU*) y *Perif.* mientras dura transfer. no pudiendo hacer estos otras tareas.
 - Mejorado por IOP (ver *Canal* abajo). Ver Fig. 20.13. Ej: *IOMMU* para *Perif.* (ver *Video RAM*)
 - Debe cuidar la *Coher. Cache* con técnicas como *NUMA*

Ver *Chip P. Sur.*

20.8 Mejoras a DMA: IOP (procesador) y canal de E/S

- **IOP (I/O Proc.):** <https://tareasuniversitarias.com/procesadores-de-es-tipos-y-estructuras.html> es una extensión del *DMA* en la que se libera totalmente a *CPU* y además el IOP posee una región de *MP* con parte del programa de I/O (llamado Progr. de Canal). Parece que cuando son varios se habla de *Canal I/O*. Ver Fig. 20.13.

- *Canal de I/O* (no confundir con Double Channel *RAM*): hoy *Arq.* en sist. de Alto *Rend.* como *Mainframes* que conecta *CPU-MP* con muchos *Dispos..* Cuando termina envia *Interr.* A veces llamado *Controlador I/O* o *Proc. I/O* o *Sincr. I/O*. Ej: *DMA*. Distinguimos dos tipos (Fig. 20.13):
 - C. *Selector*: gestiona varios de muy alta *Vel.* pero solo 1 a la vez. Ej: *Disco*
 - C. *Multiplex.*: gestiona varios lentes pero a la vez. Ej: *Impr.*

20.9 Planificación de disco HDD

- *Plan. Disco* [Plan. *I/O*]: para *HDD* dada una *Cola de Petic.* (ver *Metr.* abajo)
 - *T. Resp.* o *Vel. Acces.* o *Latencia*: depende de los t. (mseg):
 - a) *Seek (Busq.)*: (t. en ir *Brazo* de pista a pista)+(t. de cabeza en ir a sector a R/W)+(t. en rotar *Disco* al sector)
 - b) *Transfer.* (t. del copiar sector en *Buffer* de *I/O*).
- *Alg. Discipl. Colas*:
 - FIFO: V: *Rend.* alto si Petic. en sectores agrupados
 - LIFO: V: pocos mov. I: *Inanic.* si cola larga.
 - SSTF (Shortest seek time first): elige Petic. que requiera menor mov. de brazo desde su posic. actual. V: mínimo t. *Busq.* I: no garantizada t.??
- *Alg. Brazo* o del **Ascensor** o **SCAN** ([Elevator Alg.]): es una *Plan. Disco* donde el *Brazo Robot* de este se mueve lo menos posible, concretamente en 1 sentido (**arriba**) resolviendo todas las *Petic.* que encuentra hasta que llega a última pista y cambia de sentido (**abajo**).

V: previene de *Inanic.* que tienen los clásicos de antes (salvo FIFO).
I: Favorece a trabajos con pistas cercanas a cilindros interiores y exteriores (Sol: C-SCAN). Favorece a últimos trabajos en llegar como LIFO (Sol: N-SCAN).

 - C-SCAN (Circular): no cambia de sentido al terminar (siempre en misma direc.). V: *T. Resp. Equil.* a todas Petic.
 - N-SCAN (N pasos o *Colas*): se parte cola en *subColas* cada una de long. N(=2) y procesadas con SCAN (Ej: Fig. 21.13 143 a 18) durante el proc. de una cola, si llega nueva Petic. esta se mete en otra cola. Si N grande tiende a SCAN, si N pequeño tiende a FIFO. V: bajo T. Resp.
 - F-SCAN (First and Second *Colas*): emplea 2 colas, First (llena, cerrada y que se va atendiendo con SCAN) y Second (abierta a nuevas Petic.). Cuando termina con F empieza S. T. Espera max el de tamaño cola.
 - L-SCAN (LOOK): mejoras en las que se cambia Direc. cuando \nexists Petic.

Ej. 9.6 Supongamos un disco de 22 y que su cabeza está inicialmente en cilindro 11 y se solicitan los cilindros 21, 6, 13, 5, 10, según Alg. tenemos: FCFS 11, 21, 6, 13, 5, 10, SSTF 11, 10, 13, 6, 5, 21, Ascensor *Brazo* 11, 10, 6, 5, 13, 21.

Sean las pistas solicitadas de un *Disco de Brazo* móvil con 200 pistas, numeradas de 0 a 199, en el orden, 122, 90, 160, 24, 102, 89, 143, 18, 67. Para $N = 2$, las pistas solicitadas se dividen en las siguientes subcolecciones... ver Fig. 21.13. <https://www.electronicsmind.com/n-step-scan-disk-scheduling-algorithm/>

20.10 Planificación de unidad SSD

- *Plan. SSD*: las R son rápidas pero las W requieren antes tener “*Trimado*” el sector que es un proceso lento (*Trim*). Le basta FIFO (uniendo las W contiguas en mismo sector, *Secuenc. TSist. Arch.??*) pues no *Brazo* como *Disco* pero si Sectores.
 - Comando *Trim* (recortar): *Filos. Petic.*, permite al *Sist. Arch.* del SO comunicar al *Driver* del SSD que *Bloq.* están *Libre* (marcar con una *Masc.* como Tabla *Gest. Mem.*). Cuando el SSD no está ocupado tranquilamente el Recolector de *Basura* del Driver limpia dichos bloques para futuras W. Es *Unif.*. Ver *Asign., Acces. ISAM*.

20.11 Planificación de procesos con E/S

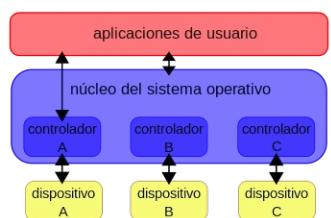
Poner ej. *Plan. Proc. con Perif.*

20.12 Gestión E/S en Linux y Windows

- *Linux: Arch. Dispos.*: Perif.=Arch. Orientados a *Block*, Carácter, *Virt.* y *VFS* (**puede gestionarlos**)
- *Win*: RESUMEN ver negrita:
 - **Plug and Play** (no confundir con *Hot Plug*): *Estand.* introducido en 1995 *Hist. SO* para numerar y Recon. **muchas variedades de Dispositivos** en arranque (*Rest.*)
 - **Obj. Manager** (Ob): *Estr. Dat.* de Win NT, que **representa a todos los Recursos, similar a Active Dir.** (*Servic. Dir.*): *Perif.*, *CPU*, *Driver*, *Fich.*, entradas del *Registr.* o incluso *Proc.* en ejecución y **de Concurr. (Hilos, Semaforos)**. **Comunicados mediante Petic. IRP** (sobre todo *CPU-Driver-Perif.*) en modo *Kernel*. Puede lanzar: *Mutex (Excl.)*,
 - **IRP (I/O Request Paq., Paquete de Petic. de I/O, no confundir con IRQ Interr.)**: es una *Estr. Dat.* de Win NT (ahora) y WDM (antiguo abajo) que usan **modo Kernel** los *Drivers* para comunicarse entre ellos y con el SO (similar al *IPC??*).
 - * En lugar de pasar un gran número de pequeños Arg. (como la Direc./Tamaño *Buffer*, tipo de F. de E/S, etc.) a un *Driver*, todos estos parámetros se pasan a través de un único *Punt.* a esta *Estr. Dat. Persist..*
 - Admin. de I/O (no confundir con *Comando devmgmt.msc??*): organiza las IRP en *Capas*. según *Llam.* (como Active Dir. Servic. Dir.??) y exporta *Subrut.* para que los *Drivers* las usen en modo *Kernel*.

- Modos Sincr. (*Bloq.* uea el *Proc.* hasta terminar *Transf.*) y ASincr. (el más usado y *Efic.* si se permite, pero implica chequear *Esper.* Activa)
- WDM (Windows Driver Model): usa *Petic.* IRP. Fué usado hasta Win2000 y reemplazado por Win NT *Hist.* *Win* para facilitar a *Desarrolladores de Drivers*

Petición de E/S



Llamada de E/S, formateo de E/S; spooling;

Asignación de nombres, protección, bloqueo, buffering, asignación de dispositivos

Asignación de valores a los registros de dispositivo, comprobación del estado

Reactivación del manejador cuando se completa la E/S

Realización de la E/S

Apl.

Programa C

Bibl.

APIs de PortAudio

Kernel (con ALSA)

WinMM

(Driver de tarj.)

DX

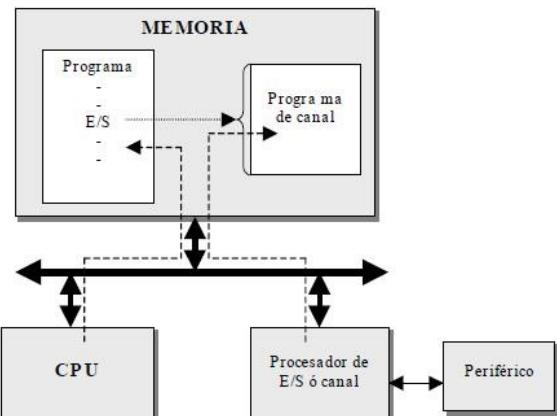
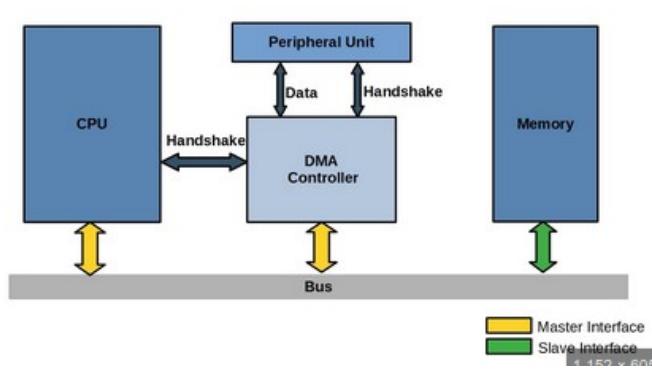
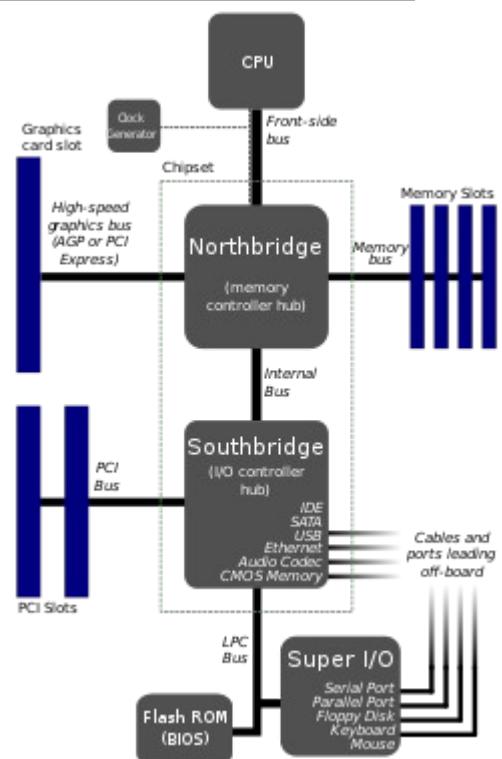
ALSA

(UnixOSS)

Otras APIs de Audio

Hardw. (tarj.)

PCI Slots



Data and address channel to main memory

Control signal path to CPU

Canal I/O: selector o multiplex

Data and address channel to main memory

Control signal path to CPU

I/O Controller

I/O Controller

0	18	24	67	89	90	102	122	143	160	199
subqueue 1 = {122, 90}										
subqueue 2 = {160, 24}										
subqueue 3 = {102, 89}										
subqueue 4 = {143, 18}										
subqueue 5 = {67}										

N-SCAN

Chapter 21

*19. Sistemas operativos: Gestión de archivos y dispositivos.

21.1 Introducción. Conclusión

- Ej: ¿Como seria la Busq. Vel. sin Sist. Arch.? Sol *INode* rapidez
- Ej-motivador: ¿qué tienen las carpetas */etc* o */dev*? Sol. el *Sist. Arch.* lo def.
- *Hist. SO*: *INode Ritchie (C)*, *Journal (IBM, NTFS 93 y Ext4)*
- Futuro: *Blockchain*, gestión en la *Nube*, *IoT*.

21.2 Sistema operativo

- *SO*: Fig. *API* vs *ABI*, *Dispos.* de *I/O*.
- *Servic.*: *Arq. Kernel* $3+1 = (\text{Mem. Virt.}, \text{Plan. Proc.}, \text{IPC}) + \text{I/O} \supset \text{Arch.}$ (este tema).

21.3 Fichero y directorio

- *Fich.* o *Arch.*: conj. *Dat.* con *Id.* Rel. con otros por *SO*. Con *Atrib.* (*Permiso*, *Tip.*, *Fecha..*)
- *Dir.* o Carpeta: en *Linux* son un tipo de *Fich.*, i.e. una *Lista* de *INodes*. Organizan *Fich Jerarq.* (algunos *SO* son planos).
 - *Direc.* (*Rut.*, o *Path*): forman un *Esp.* o *Topos* (lugar donde estás). Habitualmente con una *Tuplabarra* normal / (*Linux*) o invertida (*Win*). (como *IP*).
 - *Admin. Arch.*: *Naveg.* *MS*
 - Nombre: puede llevar *Meta* datos de *Vers.* *Fecha* (*BackUp*). Como en *Git* o *Google Drive* (*Nube*)??
 - Ej. *Populares*: */dev* (*Arch. Dispos.*), */etc* (*Etc*).

Es *Unif.* por *Busq.* Binar.=*Topos*. Ver *Servic.* *Dir.*, *INode*, *Sist. Arch.*, *Descr. Arch.*, *Comando du,df*.

21.4 Sistema de Archivos (SA)

- **Sist. Arch.** : método y *Estr. Dat.* que usa el *SO* para controlar como los *Fich.* (*Dat.*) son *Almac.* y *Busq. Efic.* (*Retrieve*) de la *MS*. Sin el, sería una larga lista de *Etiqu.-Dat.* Cada *Fich.* tiene una *Id.* (*Unic.*), organizados en *Arbol* (en *Dir.*) y *Naveg.* con *Admin. Arch.* Gestiona lo siguiente:
 - *Form. Instal.*: *Part.=crear Sist. Arch., Swap... Esp. Max.: abajo, Cuota de Disco y tam. Max. Arch. (FAT vs NTFS...).*
 - *Asign. Busq.*: abajo (*Tablas de Ini/Fin*)
 - *Segur.:* *Permisos de Acces.: abajo (Compart. en Grup.) Integr.idad de Arch.: abajo (Segur. a Err. con Redund..)*
 - *Arch. Dispos.:* abajo (tratar como *Arch. Perif.*)
 - *Tip.:*
 -) Orientados a bytes (NTFS, *INode*) o *Registr. (SAM)*
 -) *Journal*
 -) *NFS* (Sist. Arch. Distrib. *Almac. Distrib.*)
 -) *VFS*

Ver *Arbol B+, Virt. SO, BD, Esp. Nombres (Ambito), Imag. Disco, SAM, VFS.*

21.5 Características de los SA: Integridad, permisos y redundancia

- **Journal** (o *Registr.* de Bitácora): ir guardando cambios de un *Fich.*, aún no *Commit* (acometidos, *Transac.*), en un *Fich. Log.* llamado *J.* (*Registr. circular*) y *Actual.* en *Sist. Arch.* cuando se pueda (Filos. *Petic.*). *Master actual* ($Ma_{actual} = Ma_{anterior} + J$). Esto mejora *Persist.* y permite volver a *Vers.* anteriores **en caso Fall.** De *IBM* luego *NTFS 93* y *Ext3 01 Hist. SO*. Ver *Acces. Secuenc. (SAM), Vers., Loggin (LogFile, Fich.Registr. Autent.).*
- **Redund.** RAID: *Transpar.*, R. 0/1/5. En *Servid.*
- **Integr.idad:** RESUMIR: el *Sist. Arch.* debe brindar *Segur.* frente a cambios del *Arch.* intencionados (*Ataq.*) o no. Mecanismos:
 - *Permisos:* ver abajo
 - *Redund.:* *Backup* (en la *Nube*) o *RAID* (ver abajo)
 - *Herram.:* de recuperación *Journal* o con *Comandos* como *fsck, scandisk* (file system check)
- **Permiso Acces.:** a *Recursos* (como *Fich. Compart.*) para brindar *Segur. Integr.* frente a acciones intencionadas (*Ataq.*) o no (*Err.*). Se *Config.* con las siguientes *Tuplas de Autent.:*
 - P. *Usr: Usr+Grup.+Otros (ugo).*
 - P. *Op.:* r (abrir, leer, *Busq.*), w (crear, escribir, eliminar, renombrar, cambiar atrib.), x (*Ejec., Script*)

Ver Admin., DCL, Gest. Mem.. Arch. Dispos., Disco Floppy, ACL Autent., Req. (Casos de Uso)

21.6 Gestión del espacio: asignación y espacio libre (cuota de disco)

RESUMEN: Sist. Arch. controla *Cuota de Usr.* También la *Gest. Mem. Asign.* y *Esp. Libre (Trim-Basura)* de 3 formas (2x3):

- a) *Masc.* o *Map.* (Vel. si Secuenc. *Trim*)
- b) *Lista Enl.* (como *FAT INode??*)
- c) *Ind.* (*INode ISAM*). Es similar métodos de *Acces..*

- *Cuota* o Límite de *Disco*: el *Admin.* de Sist. mediante el *Sist. Arch. Asign.* a cada *Usr* una porción máx del *Esp. Libre*. Ej: en un *Web Hosting* en la *Nube*. Ver *Monit.*, *MS. Part.*
- *Asign.* en *Sist. Arch.*: se *Registr.* en una *Tabla (FAT???)*. Hay varios modos:
 - A. *Secuenc.* (similar a *Masc.* abajo): o de *Bloq.* contiguos. V: fácil de *Impl.* (basta *Direc. Inicio+Long.* para *Busq.*). I: si *Dinam.* (al inicio no sabemos *Long.* \Rightarrow *Punt. malloc*) da *Frags.* Externa
 - A. *Lista Enl.*: o *Bloq.* discontiguos. V: *Dinam.* (crecimiento indefinido como *Fich.*), no da *Frags..* I: No facil *Acces.* y *Gest. Mem.* compleja. Ej: *Blockchain* o *FAT*
 - A. *Ind.exada*: ver *INode* abajo
- *Asign.* de *Esp. Libre*: además de la *Tabla* o *Registr.* de *Arch.* (*FAT???*) el SO debe tener una Tabla de *Bloq.* sin Asignar. Hay dos formas.
 - A. *Secuenc.* o *Masc.* (o de *Map.* Bits): de Bloq. Libres 011100000110..... V: localiz. Vel. de Bloq. Contiguos y ocupa poco por lo que puede mantenerse en *RAM*. Ver *Registr.* (Estr. Log. *Fich.*).
 - A. *Lista Enl.*: de *Punt.* de bloq. libres que enlazan a otros (1º libre apunta al 2º y este al 3º..). V: *Dinam.* útil si disco muy lleno, si no ocupa mucho. Como *FAT*.
 - A. *Ind.exación*: trata el esp. libre como si fuera un fichero y con *Tabla* donde cada *Etiq.* o Llave o *Key Ind.ica* bloque libre.
 - Otras A.: *Trim SSD* (abajo), *Basura*

21.7 POSIX: INode y enlaces

- *INode* : *Estr. Dat.* de los SO *Unix* que describe un obj. del *Sist. Arch.* como un *Arch.* (con *Tabla* abajo) o *Dir.* (=Listas de INodes). Por *Ritchie (C)*, *Hist. Linux*). Es un ej. *Lista Enl.* y una de las 4 *Tablas* básicas de los *SO*.
 - *Tabla* de INodes o *Ind.*: *Block* de *128dir.* o *Punt.eros* (ver ej. abajo y Fig. 21.13). Cada Punt. apunta a un *Block* memoria (creo *2KB [inode]*). Posibilita arch. *Masiv.* por *Lista Enl..* Con *Arbol B+* se evita tener Tablas Inode de tamaño fijo.

- NOTA: *FAT* usa Tablas en localización central e *INode* tablas esparcidas por el *Sist. Arch.*.
- Ventajas: *Acces.* tanto *Secuenc.* como *Directo?*, no *Fragn.* y permite *Ln duro* (además de simbólico, varios nombres..) y *Cooper..*
- Desventajas: pequeño desperdicio de Mem. por Indirecciones.

Ver *Ln, PCB, Arch. Dispos.*, *Ext4 (Sist. Arch.)*, *Descr. Arch..*

- *Tabla Inicial o INode* en si: cada Fich. posee una de tamaño fijo con la siguiente *Estr.:* Primero se Almac. sus *Atrib.* (nombre *Id.*, *Descr.*, sacados con *Comando stat*), Luego 12 *Punt.* que apuntan **Directamente** a *Bloq.* del Arch. Finalmente 3 Punt. de **Indirección Simple**, Doble y Triple que permiten crecer *Expon. Recurs.:*
 - Simple: el Punt. apunta a un bloque de Punt., los cuales apuntan a bloques de datos del archivo.
 - Doble: apunta a Simples.
 - Triple: apunta a Dobles.

Ver Fig. 21.13

Los *Block* de Dat. también almac. los indirecciones. por lo tanto si el Block es de 512B y estamos con un PC de 32b = 4B/dir. en cada Block entran $512/4 = 128$ dir. Punt.

- *Comando Ln* : RESUMIR para crear *Enl.* a *Fich..* Es como un *Punt..* Distinguimos:
 - Duro o Hard Link: posible en *Unix* gracias al *INode* (varios varios nombres, ver arriba, *Win* no).
 - Blando o Soft o *Simbol.*: posible en *Win* (*Acces. Directo o Aleat.*).
 - Ventaja E. Duro: Fich. Varios Nombres: o *Id.*¹!
 - Ventaja E. Duro: *Arch. Compart.* o *Cooper.* (no confundir con *Almac. Distrib.*): importante mantener *Coher..*

21.8 POSIX: Descriptor de archivo

- *Descr. Arch.* (*Descr. de Arch.* o *FilDes*): nº *Id. Unic.* usado por un *Proc.* (o *Manej.*) de *Fich..* Creo es el similar al *Punt.* del *fopen* de *C Stdio*. Está en *Tabla D. A.* (abajo).
 - \forall *Proc. POSIX* \exists 3 D. A. 0=Stdin,1=Stdout,2=Stderr (salvo para los *Daemons*). Como *Arch. Dispos.*
 - *Comando lsof*: lista D. A. abiertos. Se acceden por o */dev/pts/1* (*Arch. Dispos.*) por */proc/PID/fd/0,1,2* donde PID es el de *PCB* y donde 0=S tdn,1=Stdout,2=Stderr por *Estand. POSIX* (ver *Bash* y *C*). Es una *Etiq.* o Llave o *Key* a una Estr. Dat. residente en el *Kernel*, que contiene detalles de todos los archivos abiertos

¹Si varios nombres tienen un enlace duro con el mismo nodo, los nombres son equivalentes; es decir, el primero que se crea no tiene ningún estatus especial. A diferencia de los enlaces simbólicos, que dependen del nombre original, no del inodo (número).

- *Bibl.* D. A.: *uniStd.h* y *fcntl.h* con F. como *Pipe*, *Socket*, (ver *POSIX RT*).
- *Tabla D. A.:* N°D.A. + Detalles (Archivo Abierto para *r/w/x*) Fig. 21.13. Esta apunta a *INode* y está en el *Kernel*. (parte del *INode*). Similar al *PCB* es una de las 4 *Tablas* básicas de *SO*.

21.9 Ejemplos SA

21.9.1 SA Windows

- **FAT** (File Allocation Table, *Tabla de Asign.* de Arch.): usa *Lista Enl.* (*UdBlock = Block + Punt* ver ej. abajo e *INode*). Para Win MS-DOS, su simpleza da *Fragm.* (pérdidas). Cerrado (no *Libre* como NTFS). Mejoras FAT/16/32.
 - No tiene directivas de *Segur.* (i.e. todo los *Usr* pueden acceder a todas las *Part.*)
 - Tabla de *Asign.*: guardada en *MS* donde cada *Etiq.* (Llave *Key* o Clave) o entrada *Correspond.* a un *Block*.

Ej. 9.7 FAT16 para *Disco < 2GB* (muy poco!) con Ud. de *Asign.* de *32KB* donde cada Dir. del *Block* se da con *2B* ej. A01B. Si *Cabecera Block 0000* (Block vacío), *FFFF* (último), *FFF7* (deteriorado, *Err.*). FAT32 Disc. < *2TB* con Ud. de *4KB* (menores luego aprovecha mejor Esp.)

- **NTFS** (New Technology File System, no confundir con *NFS* (*Almac. Distrib.*, ni *NFT Blockchain*)): S.A. de Win (NT Hist. SO). Organiza en Volumenes *C;D;..*
 - Usa *Journal*.
 - Usa *Arbol B+* para *Busq. Efic. Masiv.* (como EXT4).
 - Más *Robust.* y *Segur.* que *FAT*, permite *Part.* mucho mayores, Fich. gigantes, da *Robust.* y *Segur.*.
 - Cod. Cerrado (no *Libre*) luego se sabe algo por Ing. *Inv.*
- **ReFS** (Resilient File System): más *Actual.* de Win. *Escal.* mejor Fich. *Masiv.*

21.9.2 SA Unix

- **EXT (Exten.ded File Sytem):** EXT3 y 4 (mejoran EXT2) usan *Journal* y *Arbol B+* (una variación) como NTFS. Se basa en *INode* (ver abajo). Organiza los *Dispos.* como */dev/hda1,hdb2,cdrom,usb,....*
 - XFS: de 64bits altamente *Escal.*, para Alto *Rend.* (*Mainframe*), *Robust.* y arch. *Masiv.* en un solo *Host*.

21.9.3 Otros SA

- *MacOS*: APFS (el *Actual.*, optimizado para *SSD* y otras flash) reemplaza al HFS+ (antiguo).
- *IBM*: VSAM
- *ISO 9660*: S. A. para CD-ROM

21.9.4 SA virtual y en red: NFS y VFS

- *Almac. Distrib.:* *Mount*: *Samba* (vs *Libre* del SMB/CIFS) o *NFS* (no confundir con *SSHFS*).
- *VFS* (*Sist. Arch. Virt.*, no confundir con *NFS*): *Capa Abstr.* añadida encima de un *Sist. Arch.* para *Mount* otro S. A. y *Naveg. Transpar.*. Ej: SSHFS basado en **FUSE** (Filesystem in Userspace). ⊂ *Virt.* del *Almac.* esto, RAID (*Redund.*). Puede Gest. *I/O* (*Arch. Dispos.*)
² Ver *Almac. Distrib.*.

21.10 Gestión de dispositivos

- *Perif.=Dispos.* de *I/O* (comun. con exterior)
- Perif. por Uso: a) *Compart.* b) *Excl.* y c) *Virt.* (*Spooling*).
- Gest. *Linux*: *Arch. Dispos.* *ls -l /dev* (b,s y c). Uso de *Buffer* y *DMA*
- Gest. *Win*: *Obj. Manager*, *IRP*,

21.11 POSIX: Archivos de dispositivos: orientados a bloque o carácter

- RESUMEN: *Arch. Dispos.* permite a *Unixs* tratar *Perif.* con *Permisos* de Fich e interactuarlos con *Comandos read/write*. Parte de los 7 Fich. A veces manejados desde */dev* mediante el *VFS* (ver arriba). El Comando *ls -l /dev* permite distinguir A.D.O.:
Block: que si usa *Buffer* (=Cache de Disco=porción de *RAM*) (como *HDD*).
Caracteres: que no usa *Buffer* (como el *ttyACM0*)
Socket: printer
Pseudodisp o Virt.: *Descr. Arch.* */dev/fd* o */dev/null*
- *Arch. Dispos.* ([DeviceFile], no confundir con *Sist. Arch.* Orient. a *Registr.*): *Interf.* de los *Sist. Arch.* y permite tratar los *Dispos.* como *Arch.* (es una de las 4 *Tablas* básicas de *SO*). Añade la ventaja de tener las misma reglas de *Permisos* y *Segur.* de los *Fich...*.
Abstacta y oculta sus detalles Hardw. (*Transpar.*). Esto le ha permitido a *Linux* adaptarse rápido a los cambios. Se encuentran en *Dir.* */dev*.

²[Device File] En algunos sistemas tipo Unix, la mayoría de los archivos de dispositivo se gestionan como parte de un sistema de archivos virtual montado tradicionalmente en */dev*, posiblemente asociado a un *Daemon* controlador

- *Id.*: similar al *INode*, cada A.D. tiene asociado un *Driver* que tiene un Id. En caso que el mismo Driver Controle varios Dispos. se añade un SubId (*Instanc.*), Ej. *USB Serie ttyACM0* ver abajo). Pasos de *Comunic.* *OrCo* para acceder a *Subrut.* del Driver.
- Según *Linux* distingue (según *Comando ls -l /dev*): *Bloq.* (*b*), *Caract.* (*c*), *Socket* (*s*) y *DMA*. (Fig. 20.13). Como los *Descr. Arch..* Es un Desc. Arch. (*INode*)
- A. D. Orientados a *Block Mem.* (*b* según *Comando ls -l /dev*): para transm. *Paral.* de datos. Permite R/W y *Busq..* B. de tamaño fijo y conocido (1KB), se acceden a través de un *Buffer* o *Cache de Disco* (Fig. 20.13, ver), deben ser de Acces. *Aleat..* Los *Sist. Arch.* sólo se pueden montar si están en D. B.
 - Ej: *hda* (*MS*) o *HDD* (aunque también puede ser de Caracter [Wiki]!!)

Ver *Buffer*

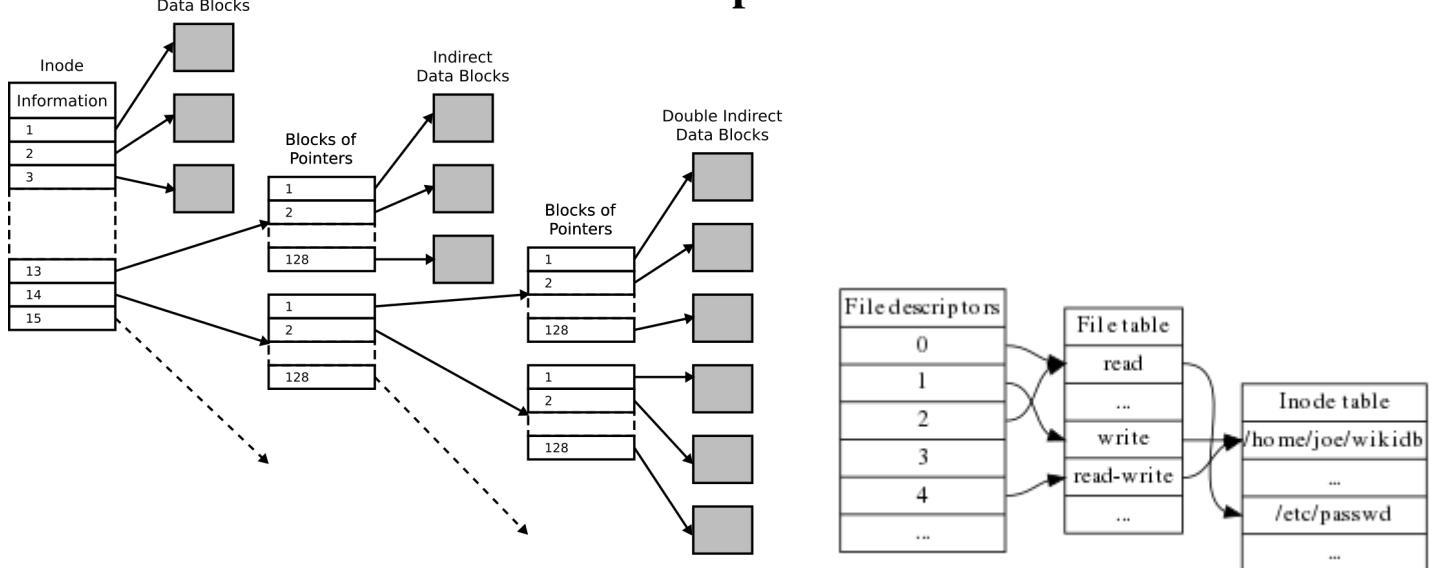
- A. D. O. Caracteres o Raw (*c*): transm. *Serie*. Permite R/W pero no *Busq.*, no se realiza almacenamiento mediante *Buffer* aunque si por *Block* alineados (nunca byte a byte o carácter a carácter, a veces se nombran *Raw Device* para evitar confusión)³. Fig. 20.13.
 - Ej: *ttyACM0* (*USB Serie de Socket Robot*) o *COM4* en *Win.* *ttyS0* (*Puerto Serie*), *ttyX* (*Terminal CLI*), *lp0* (*Puerto Paral.*), *usbdev1.1* (*USB*), *Impr.*, Ratón., *Ethernet* no.
- A. D. O. *Socket* (*s*): de *Internet*, creo A. D. O. Caracteres, **no es un Arch.** sino una comunic. entre *Proc..* Ej: */dev/printer* (*Impr.*). No confundir con crear un *Socket* (para *Robot*) con *ttyACM0* (que es Caracter).
- A. D. *Virt.*: no se comunican con D. real. Javier dice permite *Spooling* (ver *Perif.* por *Compart.*) Ej: */dev/loopX* permiten dar a un fichero la apariencia de ser un *Disco* duro para el resto del sistema. */dev/null* (acepta entrada de datos sin dar respuesta) redirecciona *Comando ls. devfsd*, VFS??).

21.12 Planificación de disco: HDD y SSD

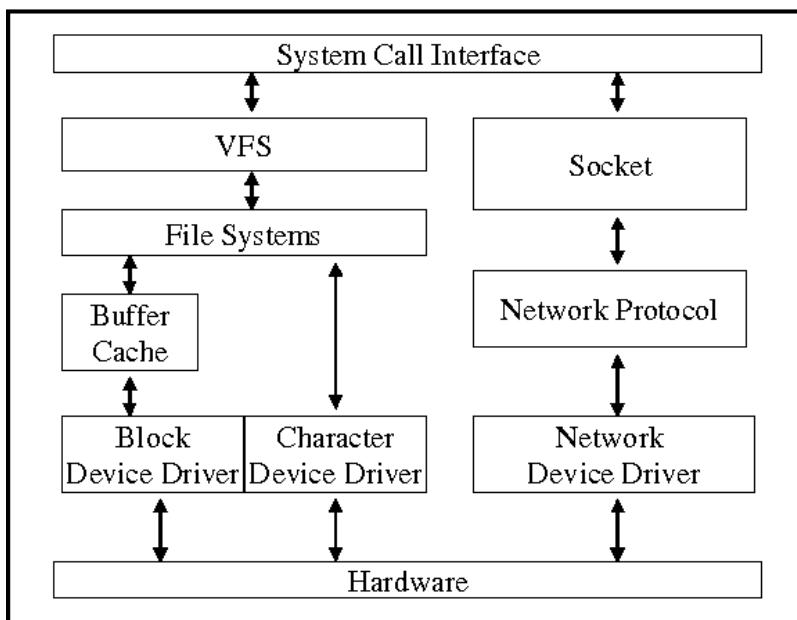
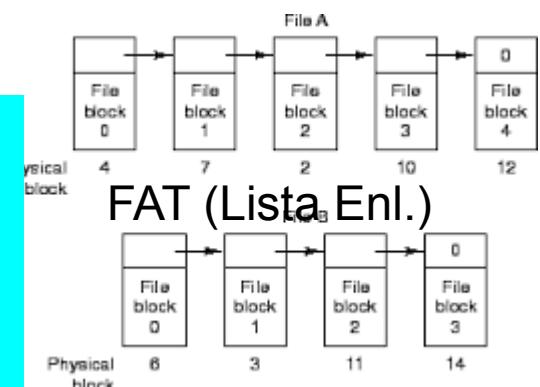
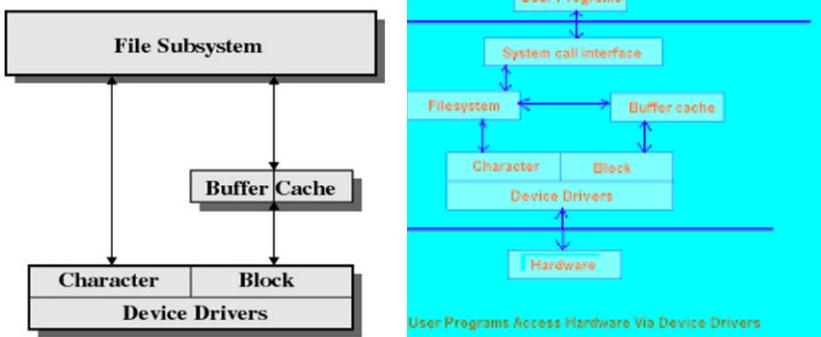
- *Plan. Disco: Cola Petic.:* Alg. FIFO, SSTF y *Brazo SCAN*. Ej: con *Metr.*
- *Plan. SSD: Trim*

³Javier dice que se usan *Cola de Caracteres*

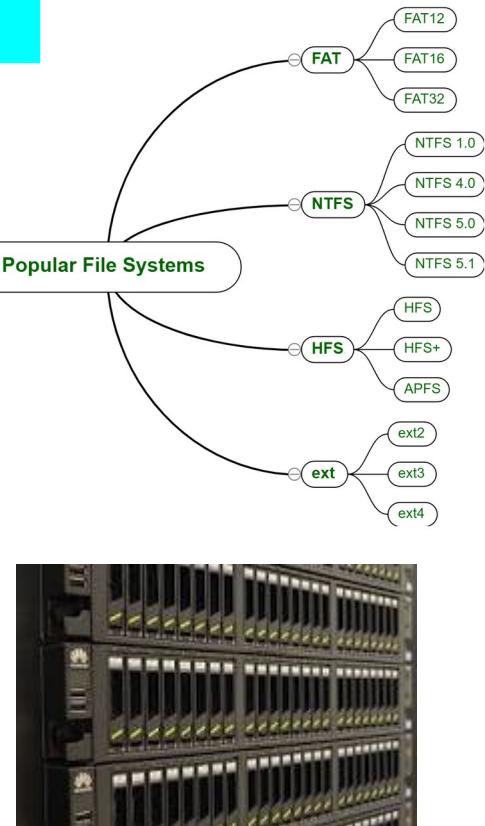
21.13 T19SOGestArchDisp



UNIX SVR4 I/O Management (cont.)



Linux I/O system



Chapter 22

21. Sistemas informáticos. Estructura física y funcional.

22.0.1 Computador. Sistema Turing completo

- *Comput. Turing Complet.*

22.0.2 Informática. Wares

- *Comput.*
 - *Informat.* : *Inform.+Autom.* se encarga de la *TAP* de la *Inform.* a gran *Vel..*
 - *TAP* : *Transm., Almac.* y *Proc.* un *Prod.*, sinónimo de *Comput..* Ver *Centro Dat., Moore.*
- Es *Unif..* Ver *Razon., Arq. Sist. I. (SO), Num.*
- *Wares:*
 - *Hardw.* : parte *Fisic.* o *Dispos.* de un *Sist..* Es un “se dice” Wittg.
 - *Softw.* : parte *Log.* o *Alg.* o *Fuente.* Ver *Apl. Web*
 - *Firmw.* (no confundir con *Driver*, ni *Firma*): *Softw.* no borrable, *Impl.ementado* y que corre en un *Dispos..* Ej. *BIOS*. Alg. implementados por H. o S.: *Firewall*. Ej: *CISCO*. *UEFI (BIOS)*
 - *Middlew.* : *Softw.* que ofrece *Servic.* a otros *Softw.* más allá de los del *SO*. En *Apl. Distrib.* son los *Servic.* entre *C. Transp.* y *API* de *C. Apl..* En *Client.-Servid.* es el “-”. Sinónimo de *Driver*. Ej: *CICS*.
 - *Malw.: Segur.*
 - *Sharew.* (ver *Libre*)

22.0.3 Alcance y potencia de sistemas

- *Distrib. /Paral.* (no confundir con *Concurr.*): *Comput.* en distinto *Esp./mismo Tmp Fisic.* de *subProbl.* para resolver un problema. Usa una *Red* (Paso *Mens.*) y permite la *TAP*

Masiv..

Ej Paral.: *Flynn, Pipeline, Palabra, GPU, Decl., Transf. FFT*

Ej Distrib.: Mod. *Client.-Servid (Middlew.) P2P Cluster*. Ver *Registr., Obj. Distrib. (Progr. Compo. CORBA) (Polimorf.)*. MIMD *Flynn, Git, Dijkstra. Almac. Distrib., Blockchain. Edge Comput. (IoT), BD Distrib., Public., Conect. Serie/P.*

- ***Vel.*** (no confundir con *Latencia*)/Tasa/Rate o Millones de Op. Por Seg. (**PS**): *Metr.* para *TAP Masiv.* gracias a un *Reloj Digit.*

- *Transm.:* Mbps (bits, bitrate)
- *Almac.:* MTPS (Transfer.). Ver *Latencia CAS*
- *Proc.:* MIPS/MFLOPS/MVPS (*Instr.*)

Ver *Ind. BD*

- *Comput.* Alto ***Rend.*** (HPC, no confundir con Alta *Disponib.*, ni *Render*): para *Cient.. Clasificación de Dispos.* por *Pot.* o *Vel.*

- *MicroControl., Embeb. Movil, PDA, Tablet, PC.*
- ***Estac. Trabajo***: PC un poco más Potent. *Comput.* para propósitos técnicos (ed. Video, Audio) y científicos. Ver *Solaris*
- ***Servid.:***
 - ***Mainframe*** : casi un *Supercomput..* Los *Servid.* suele usar sus servic. Llevan *Canal de I/O*. Ej: *IBM (z/OS y OS/360 SAM) CICS*. Ver *Met. Acces. (SAM..)*
 - ***Supercomput.*** : *Comput.* no *Distrib.* capaz de ejecutar FLOPS *Masiv.* (ver Tab. *Benchmark TOP500*). Algunos *Hist.* A. son: Atlas *Manchester (1º Pagin.)*, *HP Cray71-22* (predecir tiempo, *MicroProc. Vect. SIMD (Flynn, Array)..*), *IBM Deep-Blue96* (ganó a Kasparov), *IBM Watson11* (concurso conocimientos con *BD*), *Empresa DeepMind AlphaGo15* (Go, no es S.)
 - ***Cluster*** (no confundir con *Grid* o *Aprend. Autom.*): conj. de nodos *CPU* o *Comput.* en *LAN* que se percibe *Transpar.* (como uno solo) y hacen la misma tarea. Es un MIMD *Flynn*. Ej: *SQL Apache Spark (BD Distrib.)* Ej: Sun Microsystems *Solaris Cluster*
 - * *SO Distrib.* (no confundir con *SO Servid.*): que da esa *Transpar..* Ej: *Solaris-mc* o *QNX* o *Kubernetes* y *NAS,SAN Almac. Distrib.*).
 - * *Portab.* Batch System *PBS (Lote)*: *qsub/qdel/qstat/qhold* son comandos estandar (como *POSIX*) que lo llevan Clusters como *TORQUE*. Es un Job Schedulling o Batch-Queuing System para lanzar el mismo *Script* con distintivos parámetros I, o *Sincr.* trabajos (*IPC*). Ej: Oracle Grid Engine (antes Sun Grid Engine, *SGE*). *IBM* usa el *Job Control Language (JCL)* para su *SO z/OS*
 - ***Grid*** : tipo de *Comput. Distrib.* en el que a diferencia del *Cluster* cada Nodo puede hacer tareas distintas. Ej: *FoldingAtHome* o *SetiAtHome*.
 - *Comput. Nube: Amazon AWS.*
 - *Centro Dat.* (ver).

Ver *Palabra, Cond. Carr., Benchmark, Luz, QoS, Traf., Flynn, Buffer, Cache, Ind., Benchmark, Efic.. xFS (Sist. Arch.), Canal de I/O*

22.0.4 DSP y embebidos

- **DSP** (*Digit. Signal Proc.*): usan A. *Harvard* Modif (supongo *Turing* completo por *ALU Recurs. Chomsky*). Ej: Texas TMS32010. Hacen rápido *Transf. FFT* o convol. Suelen ser *RT*. Ej: Auricular Cancelador de *Ruido* (genera *Audio* desfasado).
- **Embeb.** o Empotrado o *Insert.*: usan *Microcontrolador* con P. *Concurr. L. C Exten. Harvard, L. ADA*. Con *Esper.* Activa+Interr.. Ej: *Micros ARM*. Ver *Win IoT, Bus Sist. Harvard*
 - *MicroControlador*: ej. *Arduino, Intel 8051* para *Embeb.* y *Motorola M68HC11 [Mecatronica]*
 - SO: *Android*

Ver *Encapsul., Container, Web E., ABI (API), PLC Autom.*

Chapter 23

Común Alg

- **Turing Comput.**: una *F.*, *Secuenc.* (ver *Concurr.*) u Obj. *Geom.* lo es si es salida de una MT que se detiene. Ej: π (Pi) si pero *Omega* no.
 - T. *Completo*.: L. T. C. (ver *Chomsky*). *Comunic.* T. C. (ver *Manej.*, *Concurr.*).
 - MT (Maq. T.): ej. *Autom.* estados Finito. *Descri.* un *Alg.*.
 - MT Universal: emula a cualquier otra MT, i.e. corre un L. T. *Completo*. (*Chomsky*). Hay muchas M. T. según Equiv. *Wolfram* (*Maq. Virt.*), *Wirth* (*Modul.*)
 - Tesis Church-Turing: todo *Alg. Equiv.* a MT \iff una *F.* es *Lamb. Comput.* \iff Turing C. (para) \iff *Recurs. General.* Bastaría encontrar un *Alg.* que no sea *Implementable* en una MT. Es un límite *Wittg.*
 - EntscheidungsProbl. o de la Parada: probl. *Filos.* resuelto por T.
 - Patrón de T.: en *Biolog.* o *Bioinform.* ej: huellas dactilares [ElPais]
 - T. Awards:
 - A.:
 - SO: *Dijkstra, Thompson y Ritchie (Unix)*
 - L.: *Kay70, Knuth (BigO), Ritchie (C y Unix), Hoare (Ord., Quicksort), Karp y Cook NP, McCarthy (LISP), Floyd (Parad. Progr.)*
 - BD: *Codd70 (Mod. Rel., Chen ER no!), Bachman BD Jerarq.*
 - R.: *Metcalfe Ethernet, [Kahn& Cerf73] TCP/IP, Lee Web*

Ver *TAI, Compr., Tma Progr. Estr., Comput., Form., Enumer., DSP. Zuse (Torres)*

- *Jerarq. Chomsky* : de Poder *Expr.* de los *L.* o de capacidad de los *Autom.* de *Recon.* (*Pars.*) *FBF*. El *Ord.* de los Autom. es:
 - A. *Combin.*,
 - A. *Secuenc.* o Maq. *Est. Finito*
 - A. con *Pila*
 - A. *Turing*.
 - Poder *Expr.*: el máximo es un *L. Turing Completo*. Lo que se gana en P. E. se pierde en Metateoría (*Completo*). Es un *Wittg.* Ver *Log.1=DML*. El máx. se consigue simplemente con una *ALU Recurs.* (i.e. ir *Op.* datos con resultados *Mem.* como *DSP???*) [*BettaTechTiposLeng*]. Con C. *Analog.* Real se puede superar.

Ver Fig. 1.7. Ver *Complet.*

- *Recurs.* *Enumer.* (o *Semi Decib.*): *Secuenc.* que tiene *Alg.* para *Gener.* sus puntos (E. con \mathbb{N} por *Combin.*) pero no para *Discrim..* Un L. R. E. tiene max *Expr.* de *Turing* (Fig. 1.7). Conj. en el que su complemento se define como “lo que queda”, por tanto si un pto \notin conj. el *Autom.* comprobador se pierde en el ∞ . Ver *Mandelbrot (Analog.)*, *Estr. Dat.*, *Array*, *Mod. Direc.*, *Pas.*, *Punt.*, *Opt. Wolfram*, *Ind.*, *Struct C*, *RTP*.
- Conclusiones: “Hawking: aprender a programar si quieres descubrir los secretos del universo”

23.1 29. Utilidades para el desarrollo y prueba de programas. Compiladores. Intérpretes. Depuradores.

- *Cod. Fuente*: progr. escrito en *Texto Plano*, *Compr.* por humano. Suele tener *Cabecera* y *Main* (ver *Estr. C*).
 - *Cod. Obj.*: (cod. entendible directamente por ordenador como *Cod. Ejec.*, *Cod. Maq.* (*Ensambl.*) pero sin las *Bibl.*, Fig. 27.11).
- Ver *API*, *Preproc.*
- T. *Ejec.* (runtime, ver *Cod. Obj.*) vs T. *Compil.*: decidir la Traduc. durante o antes de la ejecución del programa. Ej: *JIT (Interpr.)*, *Enl. Dinam.*, *Polimorf.* Ver *Wolfram*
 - *Interpr.* o L. I.: Progr. que *Traduc.* (o *Compil.*) y *Ejec.* un *Fuente Instr.* a *Instr.* o por *Block* de *Instr* (a dif. del *Compil.* ver Tab.). Ej: *CLI* de *Shell*. Ej: *Python*, *Bash* o *MATLAB*, *Java* (es medio I., resuelve *Portab.*).
 - *JIT* (Just In Time, *Compil.* en T. *Ejec.*): usado en *JavaScript* mezcla I. con *Compil.*

L.	V	I
<i>Interpr.</i>	<i>Depur.</i> rápida	Relentiza bucles o <i>Llam. F.</i>
<i>Compil.</i>	Ejec. más rápida una vez <i>Traduc.</i> , más <i>Portab.</i>	Más t. <i>Traduc.</i>

- *Compil.*: programa que *Traduc.* todo (a dif. del *Interpr.* ver Tab.) el código *Fuente* (L. Nivel alto) a otro lenguaje normalmente a *Cod. Obj.* (por máquina sin *Bibl.*). Interesa *Efic.* *Cod..* T. *Ejec.* vs T. C. Ej: decide zona *Pila* de *Llam.* (ver *CODE2*). Chequeo de *Consist.* de *Tip.* gracias a *Decl.*
 - Etapas C.: (Fig. 27.11): a) *Preproc.* (*Macro*) b) *Compil.*, c) *Enl.* (ver *Bibl.*)

Ver *Concurr.*, *GCC*, *Make*, *Ensambl.*

- *Depur.* (Debugger): parte del *Desarr.* donde se aplica Met. *Cient.* (*App Inventor*). Buscar *Err.* en T. *Compil.* (fácil y automát.), en T. *Ejec.* (difícil). También *Opt.* (*Efic.* o *Eleg.*). **Ejec. Paso a Paso** permite *Ing.* *Softw. Inv.* y saber valor de var. en todo momento. Ej: GDB de *C* en modo texto y gráfico *IDE* con *breakpoints*. Gracias a *Tip.* es (gratis). Ver Tab. *Interpr.* vs *Compil.*, *Test*

- **Enl.** : etapa de *Compil.* donde el *Compil.* crea un fichero *Cod. Obj.* (.o) para cada fichero *Fuente* (.c) antes de linkarlos juntos y crear ejecutable final. Es *Reescr.* (*Vincul.* y *Subst.*), *Vincul.*
 - Ej: 1) Crear cada .o: uno a uno *GCC gcc -Wall -O3 -c serverTCP/serverembedTCP.c; gcc -Wall -O3 -c decoder/golay.c* 2) Enl. los .o con las librerías (-lm de Matem.): *gcc -o fatherTCP -Wall -O3 serverembedTCP.o decoderembed.o golay.o serverembedUDP.o RecognVAD.o recogembed.o statistic.o fatherTCP.o childUDP.o common.o -lm* 3) NOTA: para *Proyec.* con pocos *Modul.* basta incluir los .h y compilar poniendo 1º el main: *g++ main.cpp m1.cpp m2.cpp*
 - Ej: *Java* ver *Bibl.*
 - Ej: *CSS*

Ver *Ln*, *Lista E.*, *Arbol B+*, *Vincul.*, *Join*, *Enl.*, *Din.*, *POO*, *Enl.*, *Din.. C.*, *Enl.*

23.2 30. Prueba y documentación de programas. Técnicas.

- **Docum.** entar: *Texto* que ayuda al *Manten.* y debe ser *Public..* Ver *Desarr.*, *Proyec.*, *Progr. Estr.*, *IDE*, *Coment.*, *Preserv.*, *Instal.*, *Test*, *Req.*

23.3 33. Programación en lenguaje ensamblador. Instrucciones básicas. Formatos. Direccionamientos.

- L. **Ensambl.** (assembly): L. Nivel bajo con conj. *Instr.* memot. (*IEEE 694* o *CODE2*). Ej: RISC. De los más antiguos *Hist. L.*. También es el *Traduc.* de L. *Compil.* a L. Maq.. Comparado con L. Nivel bajo: Vent.: rápido *Ejec.*, aprovecha *Recursos* (menos *Mem.*). Inconv.: in*Compr.* (lento programarlo), no *Portab.* (específico de maq.), detalloso (conocer *Gest. Mem., Perif..*)
 - Grace Murray Hopper: mujer militar con gafas, 1er L. Ensambl. 1955 (antes que *Fortran*).
 - L. Maq. o *Binar*: Conj. *Instr.* en Bin. o Hexa, directamente *Ejec.* sin *Traduc..* Inconv. añadido: no *Compr.*. Ver ABI (*API*).
 - *RT*: fácil control de t. de cada *Instr.* (ver *Arduino*)
 - *GoTo*: usado en E. como *Arduino* que no tiene *F*.

Ver *Cod. Obj.*, *Mod.*, *Direc.*, *C.*

- **Instr.** o **Primit.**: Atom. de la *Semant.* o mínima Ud. de *Proc.* disponible a un programador. Ej: RISC de *CPU* o *Ensambl..* Se pueden *Compo.* y *Decl..* Distinguimos:
 - I. *I/O: Perif.* (*print,scanf*).
 - I. *Estr. Control* (*if,for*).
 - I. *Asign.*: *Var.* (*a=3.14*).

- I. Decl.: de *Obj.*
- I. TSL: para *Excl.* Mutua

Ver *Interpr., L., Alg., Progr., Pal. Reserv., Comando*

- Mod. *Direc.*: los 3: a) Direc. b) *Punt.* y c) *Offset*

Chapter 24

+23. Diseño de algoritmos. Técnicas descriptivas. (27OctJoseA)

24.1 Introducción. Conclusión

- Ej: ¿Como desarrollar? es como construir una casa (mejor no equivocarse en los cimientos)
¿Como consigue la *Efic.* Quicksort o FFT? por *Div. Venc.*
¿Qué es *Turing Complet.*?
- Hist. *Knuth* ([The Art...] *BigO*), Mills Top-Down, *Hoare* (Quicksort), Bellman53 *Progr. Dinam.*, Chaitin (*TAI*)
- Futuro: *Autocompl.*, *CASE IA*

24.2 Algoritmo vs programa

- *Alg.* (no confundir con *Progr.*): *Secuenc.* de *Instr. Ord.*, finita y no ambiguas (en un L. *Form.*, ver *Progr. Propio*) para de unos *I/O* realizar un *Comput.uto* (*Proc.* o tarea) o *Probl.*. Tiene una connotación más *Abstr.* que *Progr.* por lo que *Cod.* suele ser un *Diagr.*.
 - Ej: *Transf. FFT, RSA.*
 - Más Ej: de A. las Teor. *Conoc. Fisic.* (*TAI*). Ej: *Protoc.*, el A. más simple es una *Correspond..*

Es *Unif.* Ver *Inform., Conoc.* (Ec. Dif. *Fisic.*), *Softw. Razon., Distors.*

- *Progr.* (no confundir con *Alg.*): Fase del *Desarr.* en el que se *Cod.*, *Impl.* o *Instanc.* en un L. un *Alg.* (hecho antes en la fase *Dis.*). Un P. es un “se dice” Wittg. mientras que alg. es un “mostrado” (*Abstr.*) Fig. ??.
 - Progr. Propio: ver *Tma Progr. Estr.*
 - Usado en *Markov* (Decision Process Aprend. Autom. por Refuerzo)

Ver *Ing. Softw. Inv., MultiProgr. (Multitar.)*

- *Estr.* de un Programa

- Descr. de Dat.: Caracter. *Tip.* o *Estr. Dat.* (simple vs compuest), *Literal*, *Id.* (cte vs *Var.*).
- Descr. Acciones: *Expr.*, *Op.* e *Instr.* (los 4 tipos)
- Caracter: de buen Alg.: Memot. (RobEfi PortFia LePre ReFini)
 1. RESUMEN: Las 3 de Progr. Propio o Turing C. (*Tma P. Estr.*) + clásicos:
 - a) 1. *Unic.* punto I y O. 2. Camin. por todas las Instr. 3. No bucles Inf. (*Turing*)
 - b) *Efic.* (Esp/Tmp), *Eleg.* (Legible),
 - c) Reproducible (Met. *Cient.*),
 - d) *Portab.* (Independ. del Hardw.) IMPORTANTE: elegir buen *L*.
 2. Javier: Fiable o repetitivo: resultados exactos y precisos Repetible: otro puede *Impl.* Met. *Cient.* Preciso: no ambigüo en pasos *Robust.*: respuesta independ. de circust. donde se *Ejec.* *Portab.*: indep. del Hardw. Flexible: adaptable con facilidad a cualquier L. *Efic.*: uso de *Recurso* opt. *Compr.*: legible *Turing* parable: finito Otras: *Escal.*

24.3 Ciclo de desarrollo de software

- Ciclo *Desarr.* o Ciclo Vida (Development or Life Cycle, no confundir con *Proyec.*): en Ing. Softw. es Met. *Cient.* (ver *Agil.* abajo) tiene 5 fases (ADIVM) Fig. 24.8:

1. *Anal.*: *Req.* del *Probl.*, *Def.* I/O requeridos.
2. *Dis.* (Diseño): del *Alg.* y creación de un *Proyec.* (inicial) (*Opt.* y *Descr.*)
3. *Impl.*ementar o *Progr.* o *Cod.ing*: del *Alg.* con un *IDE*. También *Docum.*
4. *Verif.* o *Depur.* o Prueba o *Test* (*Markov* de Harlan Mills, *Calidad*): del *Progr.*
5. Despues: *Manten.* *Despl.* (*Public.*, ..), y mejora (Met. *Cient.* *Opt.*)

Ver *Eleg.*, *Plan.*, *CASE*, *SGBD*

- *Desarr. Agil.* o Incremental: es *Filos.* del “sigue leyendo si te atrancas”, ir descubrir. *Req.* conforme se dan soluciones y trabajar en equipo *Cooper.*.
añadir nuevas funcionalidades en cada *Iter.* (o *Vers.*) de un *Proyec.* *Cooper.* según surjan necesidades.
Su principal ventaja es *Vel.* en Comercialización (entrega a Usr) y mitiga riesgos. Se usa en **Hackathon** (Desarr. *Cooper.* en 24-48h).
Es Met. *Cient.*, *Evol.*, Met. Polya (*Probl.*). Se complementa con *DevOps*.
 - *Desarr. RAD* (Rapid Application Development): Interactivo con Prototipos y *CASE*.

24.4 Diseño de algoritmos: estructural, objetos e interfaces

- *Dis.* *Alg.*: parte del *Desarr.* Softw. en el que se *Descr.* el *Alg.* conforme a las especificaciones hechas en la fase de *Anal..*
Elegir *Parad. Alg.* que más convenga para *Opt. Efic.*

Descri. sin ambigüedad.

Distinguimos Top-Down y Bottom-Up (ver abajo).

- D. *Estr.* (este), D. *POO* (ver abajo), D. *Interf.* (UI vs UX, ver *GUI*).
- Otros D. *BD*: ver *Model. ER*

Ver *Probl.*, *Patr.* D., *Parad.* *Alg.. Cooper.*, *CASE*.

- D. Descend. o **Top-Down** (Deducir): similar a *Div. Venc.* (ver abajo). Se empieza con Sol. *Probl.* gorda o **Esqueleto** y luego se va **Refin.** (parece *Agil.??*) y Rellenando (si *Cooper.*..).

La progr. *Estr.* (con *C*) lo es (ver Mills *Hist. L.*). Des/Ventaj (ver Bottom-up)

- Ventajas: ver Bottom-Up y además (Javier): a) *Cooper.* (cada uno hace una parte). b) *Modul.* (*Axiom.-Ortog.* (*Cohes.*), *Reus.*,*Eleg.-legible*, mantenible..)
- Ej. SFD aplicar reglas (*Combin.* válidas en un nivel alto de lenguaje) basadas en *Model.* (top) para agrupar los fragmentos (down) que sean de *Voz.*

- D. Ascendente o **Botton-Up** (Inducir): ir resolviendo conforme aparecen problemas.

La *POO* con *Cpp* lo es porque comienza definiendo *Obj.* base que luego son *Enl.* por Obj. sup. (ej. *Arbol SPN*).

Des/Ventajas: https://asana-com.translate.goog/resources/top-down-approach?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=sc

- Ventaja: empezar pruebas tempranas de las bases y *Reus.* de Cod. a bajo nivel. Ideal para Equipos que *Cooper.* *Creat.*ivamente. El Top-Down bueno para conj. de muchos equipos pequeños *Jerarq.*..
- Desvent.: no cumplir obj. globales al tener una visión obj. final, problemas de *Interop.* y duplicidad de esfuerzos. El Top-Down no es buena para gran *Vel.* de respuesta, puede desmoralizar por limitar la *Creat.* y ralentizar la resolución del *Probl.*..
- Ej. *Recon..* Ej. en Lego empezamos con piezas bien definidas y vamos poniendo sin saber seguro el resultado emergente.

- D. *POO*: Anal. un *Probl.* de Negocio, *Docum.* y def. las *Tip. Dat. Abstr.* (Dat.+Op.).

Se usa *Diagr. UML* y Caso de usos (*Req.*).

- Pasos Anaya:
 - a) Describir problema,
 - b) Buscar objetos y acciones
 - c) Asignar atrib. y métodos
 - d) Representar todo en Diagr. UML
 - e) Implementarlo
- Des*Compo.* en Obj.: si muchos con distintas responsabilidades y compleja. Aumenta tam. del *Cod.* (no *Eleg.*) pero útil si reparto *Cooper.* (esqueletos) y futuras mejoras (*Manten.*).
 - Usar *Her.*
 - Evitar *Probl. Yo-Yo (Her.)*
- Anal. de *Req.* o *Semant.* (lectura): Nombres (*Obj.* o *Atrib.*) Verbos (*Metod.*), es subjetivo como *Model. ER*. Parte *Visual* vs *Log.* (Casos de Uso (*Req.*)))

24.5 Técnicas descriptivas de diseño

- **Descr.** : de la *Estr.* de un *Alg.* en el *Dis..* Puede ser *Form.* (*Progr.*) o *InForm.* (*PseudoCod.*, es un “se dice” *Wittg.*). 3 formas:
 - *PseudoCod.*: D. *inForm.* entre humano y *L.* Progr. Posee *Instr.* de I/O/Proc. *Estr. Control Subrut.* y *Coment.* Fig. 24.8 (poner 2 ej.). *Herram.* *PSeInt*, *CASE*
 - *Tabla de Decisión*.: es una F. *Combin.* Multivar. con I (*Condic.*) y O (Acciones) y E (*Est.*). Ej: Circ. *Combin.* o *Secuenc.* o *Turing*. Fig. 24.8
 - *Diagr.* Flujo, Nassi-Schneideman, Est. (ver abajo)

Ver *PCB*, *Tabla Pagin.*, *Model.* *BD*, *Arq.* *ANSI/SPARC*, *Arq.*, *Estr.*, *Descr.* *Arch.*, *DDL*, *Def.*, *Decl..*

- **Diagr.** : DiGraf. del flujo de *Dat.* y *Op.* para *Descr.* un *Alg.* Se puede ver como un *L.*
 - D. *Flujo*: para *Estr.*. Distinguimos:
 - a) **Ord.inograma**: para fase *Anal.* de *Desarr.* *Secuenc.* *Instr.* con *Simbol.* *ANSI*: Ini/fin, I/O, Proc., *Condic.* (dec.), *Subrut.*, *Conect.* fuera/dento pg. Ej: Fig. ??
 - b) **Organigrama**: para fase de *Progr.* ej: Programa, Perif I/O, Soporte R/W. Ver *Control Flujo Datos*
 - D. Nassi-Schneideman (NS): mezcla de D. Flujo+Pseudocód. Sin flechas representa el flujo en una caja. Específicos para Progr. *Estr.* (no permite el *GoTo*) mientras que D. Flujo más versatiles. Fig. 24.8
 - D. UML (Unified Modeling Language): varios tipos: D. *Her.* (ver) y D. Casos de Uso (*Req.* F.), *Umbrella* (*CASE*)
 - D. Gantt: *Rut.* *Crit.*
 - D. Venn: *Op.* *Log.* y *Conj.*
 - D. Estados: para *Autom.*
 - D. *Model.* *ER*.
 - D. *Map.* Conceptual: *Rel.* del *Conoc.* (solo cuadrados)

Es *Unif..* Ver D. Hasse (*Conoc.*) *Cat.*, *Autom.*, *CASE*

24.6 Diseño: técnicas para eficiencia

- **Efic.**: RESUMEN *BigO* (Tmp/Esp). Metodos de cálculo del *BigO* (teórico, empírico, híbrido). Técnicas:
- **Eleg.**: (ver, *TAI*)
- **Efic.** *Alg.* (no confundir con *Rend.* o *Eleg.*): *Metr.* *Softw.* capacidad de hacer lo mismo *Semant.* usando menos *Recursos Esp./Tmp.* (no implicando el *Opt.* Global)
No lo estudia *TAI* (que es más de *Eleg.*).
Para Calc. *BigO* 3 formas:

- Teórica:
 - a) T_{mp} (CPU): $T = \sum_i T_{ip,i} Instr.$
 - b) $Esp.$ (MP y MS): $S_{max} = \max_i \{Dat, Instr\}$
- Experimental: depende del *Micro* físico, *IDE MATLAB*
- Híbrida

Técnicas de Efic.

- “*Opt.* prematura es la raíz de todo mal” [Knuth, The Art...] *Hist. L.*, mejor solo de fragmentos críticos y por necesidad, (puede hacer menos legible)
- Opt. *Esp.* por Memoización (*Eval.*): *Mem.* en *Cache* resultados frecuentes $\pi, \cos(\pi/4), \dots (Pi)$
- Opt. *Tmp*: replantear *Alg.*, *Fact.* primos (hasta $\frac{n}{2}, \sqrt{n}, eratostenes\dots$)
- Ej: Re*Fact.* Ej. *Transf.* FFT y QuickSort Ej: E. o *Filos.* *MATLAB* (basada en *Lin.*). Ej: gracias al *Compil.* *GCC -Wall* y *-O3*, hacer lo mismo con más rapidez y menos recursos.

Ver *Alg. Voraz*, *MetaBiolog.*, *GPU*, *Lista vs Tupla*, *T. Resp.*, *Estr. Dat., Acces.*, *RISC. Ecolog.*.

24.7 Paradigmas de diseño de algoritmos

- *Parad. Alg.* de *Dis.* (no confundir con *Parad. Progr.*): es *Gener.* o *Abstr.* de conj. de *Alg.* similares:
 - *Div. Venc.*,
 - *Recurs.*: ver
 - *Progr. Dinam.*, *Alg. Voraz*, *Backtracking*.
 - *Busq.* Fuerza Bruta o *Combin.*
- *Div. Venc.* : des*Compo.* un *Alg.* en *subF.* (y estos en otros *Recurs.*) con soluciones independientes y luego recomponer.
Ventajas añadidas: *Efic.*, *Paral.*, *Cache* (buen uso), Precis. Num..
Usado en: QuickSort (*Ord.*) y FFT (*Transf.*).
Se prueba por **Inducción**. Ver D. F., *Alg. Voraz*, *Dis.* Top-Down

24.7.1 Ordenación y Busquedad

- *Ord.* (Sort, Ordenamiento):
 - Eliminación:
 - Burbuja (Bubble)
 - **Quicksort**: IMPORTANTE es un *Div. Venc..*
 - a) Tomar elemento cualquiera y resituar (permutar) a su izda todos sus menores y dcha sus mayores, el pivote ya tiene su pos. ok.
 - b) Repetir lo mismo (*Recurs.*) para cada *subLista* de izda y dcha.
 - c) $BigO([n \log(n), n^2])$ (mejor a peor caso según elijamos pivote).
- Por **Hoare** (Estr., Monit. Semaforo RT, Null Vulner., Turing A., *Hist. L.*)

- IMPORTANTE: los 3 tipos de Probl. Examen son: Quicksort, Arbol B o Lista doble enlazada o Hanoi)
- *Busq. =Consult.=Topos* (Buscar, Search o Retrieve=Recuperar en *Estr. Dat.*): se hace con *Expr. Reg.*
 - B. *Lin.* o *Secuenc.*: *BigO(n)*. Ej: *Lista* o *Acces. Secuenc.*
 - B. *Combin.*, Exhaustiva o Fuerza Bruta o por Casos: *BigO(P o NP)*. *Enumer.* todos los casos posibles y evaluarlos para localizar los que cumplan la *Condic.*. Poco *Efic.* *BigO*. Muchos como Branch and Bound (*Back*) son mejora. Ver *Estr. Dat.*, *Ataq. F. B.* (*Autent.*), *Vulner.* WPS (*WPA*).
 - B. *Binar.*: IMPORTANTE *BigO(log₂(n))*, ej ISAM (Cascada) y Factor Block (*Ind.*). Ver QuickSort (*Ord.*), Cascada=*Busq. Binar.* (*ISAM*).
 - B. *Masiv.*: Factor Block (*Ind.*), Arbol B y B+ (ver *Sist. Arch. NTFS, Index VSAM*), PageRank *Naveg.*, *Graf.* (DFS, A*)

Es un *Unif.* por B. *Vel.* *Naveg.* *Topos=Dir.*. Ver *Autom.*, *Consult.*, BFS y DFS *Graf.*, Recorrido (*Arbol*), *Mem.*, *Estr. Dat.*, *Mem. Asoc.*, *Topos*.

Usando un *Diagr. Flujo Descr. Alg.* Quicksort (SubLista Recurs.)

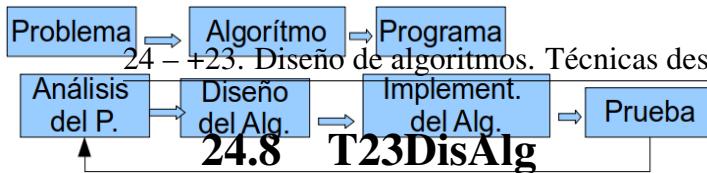
24.7.2 Paradigmas de optimización

- *Opt.* : dado un *Esp.* de sol. a un *Probl.* en forma de *F.* (*Analog.*) o de *Alg.* (*Digit.*), elegir las/la sol. que cumpla *Condic.* (local vs global). *NP* por *Wolfram* (ver)
 - Ej: de Esp./Cond. *Analog.* (*NN* o $f(x,y)/s.t. 1 < (x,y) < 3$, *Anal.*) vs *Digit.* (*Rut. Graf.*/ $x \in \{par\}$ *Dijkstra*). Malo si no *Lin.* o *NP Wolfram Enumer.*
 - *Ing. Softw.*: Opt. en *Desarr.* (*Eleg.* y *Efic.*). *Proyec.*: *Precio Recursos* de un
- Es *Unif.*. Ver *Fisic.*, *Busq. Combin.*, *Alg. Voraz*, *Back*, *SGD Aprend. Autom.*, *Huffman. U* (uso *Plan. RT*), *Compo. O.* (*Conect.*), *Opt. Consult.*
- *Alg. Voraz* : heuristica que consiste en dividir en subtareas (*Div. Venc.*) y elegir el *Opt.* local de las subtareas para obtener el opt. global o una aprox. Son muy *Efic..*
 - *Progr. Dinam.* : [Bellman53] *Hist. L.*: (*Progr. Dinam.*) *Opt.* donde podemos partir *Recurs.* el Probl. en Subprobl. locales (simples y similares *Div. Venc.*) porque se da *Markov*. Usa *Forw.-Back* y a veces *Alg. Voraz* para los locales. Se gana en *Recurso T.* (*CPU*) a cambio de más *S.* (*MP*) al guarda Tabla de opt. locales. Ej: **Alineam.** Prot. (*Bioinform.*), *Probl. Rut.* o *Viterbi* (discreto) o *C. Variaciones Geodesica* (continuo *Geom.*) o *Progr. Enteros*. Ver *DTW (Aprend. Autom.)*
 - Ej: *Alg. Fulkerson*, *Alg. Kruskal* (*Arbol Recubr. Min.*), *Alg. Dijkstra*, *Progr. Dinam.* (es una *Gener.*)
 - *Probl. Mochila (Knapsack)*: [Dantzig] *Hist. L.* propuso sol. subopt. (*Alg. Voraz*). Casos especiales son: P. Suma 0 (21 *NP-Completo*) y P. Cambio Monedas.
- *Back tracking* o Marcha Atrás: *Opt.* en un *Arbol* basado en *Busq. Profund. Recurs..* Ver *Progr. Dinam.*, *Recurs.+Pila Llam..*

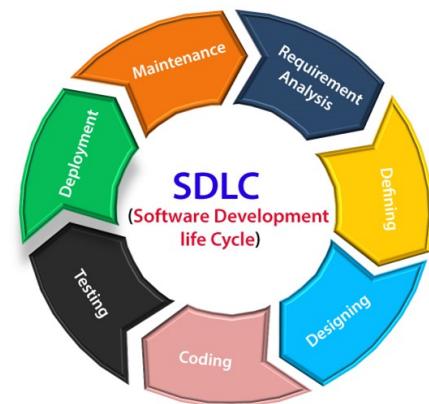
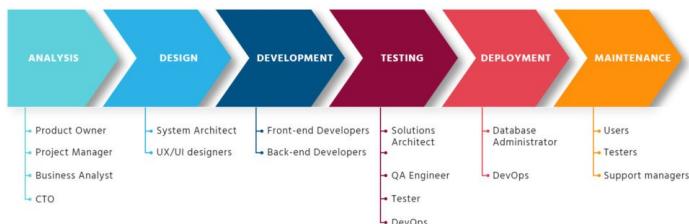
- Ej de uso: optimizar juegos como *Probl. 8 Reinas Ajedrez*.
- **Branch Bound** (Ramificación y Poda): variante del *Back* para opt. *Combin.* en un *Arbol* para Probl. *NP-Duros* o *NP* como *Rut..* Un Branch sin Bound *Equiv.* a Fuerza Bruta *Busq. Combin.*, el Bound va podando caminos no opt. y permite volver a Nod. más prometedores que el actual.

Ver Backup (*Preserv.*, *Redund.*), *Daemon*, Backbone (*Cable Estr.*), *Tracer*, Backdoor (*Malw. Exploit*)

- Alg. *Est.* y Meta-Heurísticos: Ej: *SGD*, Integración Monte-Carlo, Alg. Colonia de Hormigas



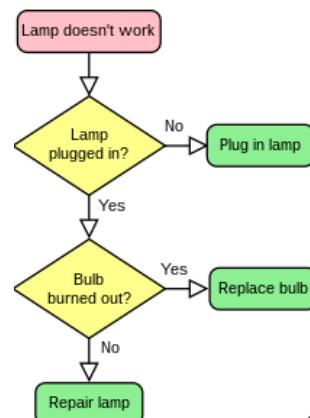
6 PHASES OF THE SOFTWARE DEVELOPMENT LIFE CYCLE



Procedimiento Ordenar (L)

```

//Comentario:  $L = (L_1, L_2, \dots, L_n)$  es una lista con  $n$  elementos/
k ← 0;
Repetir
  intercambio ← Falso;
  k ← k + 1;
  Para i ← 1 Hasta n - k Con Paso 1 Hacer
    Si  $L_i > L_{i+1}$  Entonces
      intercambiar ( $L_i, L_{i+1}$ )
      intercambio ← Verdadero;
    Fin Si
  Fin Para
Hasta Que intercambio = Falso;
Fin Procedimiento
  
```

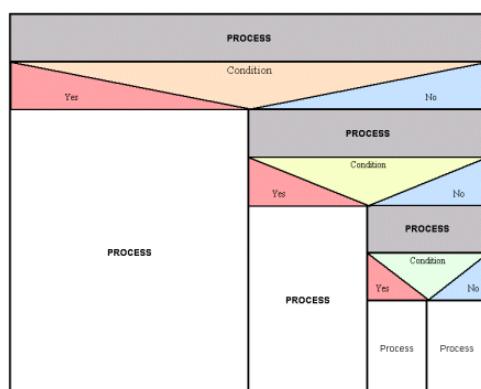


Printer troubleshooter

		Rules							
Conditions	Printer prints	No	No	No	No	Yes	Yes	Yes	Yes
	A red light is flashing	Yes	Yes	No	No	Yes	Yes	No	No
	Printer is recognized by computer	No	Yes	No	Yes	No	Yes	No	Yes
Actions	Check the power cable			✓					—
	Check the printer-computer cable	✓		✓					—
	Ensure printer software is installed	✓		✓		✓		✓	—
	Check/replace ink	✓	✓				✓		—
	Check for paper jam		✓		✓				—

ANSI/ISO Shape	Name
→	Flowline (Arrowhead) ^[15]
○	Terminal ^[14]
□	Process ^[15]
◇	Decision ^[15]
[/\]	Input/Output ^[15]
—	Annotation ^[14] (Comment) ^[15]
—	Predefined Process ^[14]
○	On-page Connector ^[14]
—	Off-page Connector ^[14]

Shape	Name
ylinder	Data File or Database
rectangle	Document
hand	Manual operation



Selection Sort

for i from 0 to n
m = i
for j from i + 1 to n
if a[j] < a[m]
if yes
m = j
else %
endif
endif
h = a[m]
a[m] = a[i]
a[i] = h

Chapter 25

+24. Lenguajes de programación. Tipos. Características. 15OctAngela

25.1 Introducción. Conclusión

- Elegir L. según tipo de *Probl.*, *Popular*, Curva Aprend...
- *Hist. L.* : 4 generaciones: *Fortran* 1º Alto nivel
- Fut.: *Progr. Fun.*

25.2 Lenguaje de programación. Chomsky

- *L.* Program.: L. *Form.* (de *FBF* o *Sintax.* clara, normalmente *Turing Complet.*) que permite *Descr.* un *Alg.*, *Comunic.* el L. Natural con L. Maq.. Gracias a un Conj. finito de *Instr.* que dan un cierto poder *Expr.* Es *Unif.*. Ver *Parad. Progr., Diagr., Simbol.*, Wittg., L. *Music.*
- *Chomsky*: A. *Combin.*, *Secuenc.*, *Pila* y *Turing*

25.2.1 Segundo cercanía a máquina

- *L. Nivel:*
 - Alto: Vent: cercano a humano (en Inglés), curva Aprend.), *Portab.*, *Depur.*. Inconv: no aprovecha *Recursos* (ocupan más memoria), *Ejec.* lenta. Ej: *Lean*, *Java*, *Python*, *LISP* y *Yushchenko (Punt.)*
 - Medio: *C. Bash??*
 - Bajo: L. Maq. y *Ensambl.*. ver Ventajas.

Ver *Compil.*

25.3 Según generación (o paradigma)

- *Hist. L.* : 4 generaciones: REVISAR No es histórico parece es por tipos (ver abajo) (comparar con *Hist. SO*):
 - 1^a: L. Maq.
 - 2^a: L. *Ensaml.* muy *Depend.* de la Maq.
 - 3^a: L. *InDepend.* de la Maq. y amigables al Progr.
Ej: Los 1º L. de alto Nivel, *Propos.* General y Espec. *Fortran*, y Basic, *COBOL*.
Ej: Casi todos los *Proced.*:
Estr. C,
POO Cpp, Java y PHP (Internet), *Visual Scratch, Bash??*
Concurr. LabVIEW, Progr. Fun. LISP,
 - 4^a (4GL): L. *Decl.*, automatizar op..
Ej: *SQL, Python* es 3º y 4º. También *IDE, Autocompl., Git, Apl. Web* de Progr.
SGBD (automat. def., consulta, informes..),
 - 5º: Resolver un *Probl.* sin el *Progr.* Ej: Mercury IA ChatGPT??

RESUMEN: *Evol. Abstr.: Estr.* (bloque inmanejable) → *Modul.* → *POO* (en la que los datos y Op. están separadas y los *Modul.* intercambian datos.). Permiten *Dis./Impl. Cooper.* (Esqueletos)

25.3.1 Paradigmas de Programación

- *Parad. Progr.* (no confundir con *Parad. Alg.*): Hay casi tantos *L.* como *Probl.* por lo que elegir el adecuado para *Dis./Impl.*. A los viejos *L.* se le añaden nuevos Parad. Ej: C a *Cpp* o *LISP* a *Common-LISP* que añaden *POO*. Termino de Robert Floyd78 (*Estr., Alg. Dijkstra, Turing A.*) basado en Kuhn. Los 2 básicos son (Fig. 25.8):

- *Imper.* : el programador instruye a la máquina cómo cambiar su estado. Destacan:
 - * *Proced.* y *Estr.*
 - * *POO*: uso de los objetos (un tipo de estructuras de datos)
- Progr. *Decl.* (L. No *Proced.*): *Parad. Progr.* opuesto a *Imper.* donde el Usr *Expr.* qué tarea hacer (con un *Predic.*) sin especificar el **cómo** (no tienen *Estr. Control*) Simplifican la *Paral.* y la mayoría son 4GL (*Hist. L.*). Destacan:
 - * *DML*: *SQL* es por ello que tiene que *Opt. Consult.*
 - * *Progr. Fun.*: Reactiva (*Event.*) o *Haskell*.
 - * *Lean* y *Constr.??* (influenciado por *Coq* y *Haskell*)
 - * *Log.*: *Expr. Reg.* y *Calc. Rel. Domin. Lacroix??*

Ver *DML, F., Var., Dinam., Var., Descr., F. Virt., D.* (Statement vs *Expr.*)

Otros son:

- Progr. *Proced.* : es un tipo de Progr. *Imper.* se programa como hacer la tarea. Se basa en llamar a *Subrut.* (o proced.). Son los 3GL (*Hist. L.*). [ERRATA??: Wiki Español dice que *Progr. Fun.* lo es]
Ej: *C* y *Fortran, SQL/PSM (Persist.)*.
Ej. No P.: *Alg. Rel.* (REVISAR) Ver *Gener. Procedural (Videojuego)*.

- **Progr. Fun.** : tipo de Progr. Decl. basada en Calc. Lamb. donde la Def. F. son Arbol de Compo. de Expr. que Map. valores en otros (Curry), mas que una Serie de Instr. Imper. que se actualizan en Ejec.. Ej. Punt. como F. en C, Haskell, LISP).

- * **Map.** , Filtr.er o Reduce [Thorsten]: elementos de Lamb. cada vez más incorporados en L. Popular [BettaTech]. Sust. for Iter.. Ver Lista, Map. BDOO Obj. Rel., Masc. bit. (Trim), Correspond., Serielizar (JSON), M. Conceptual (Conoc.), MMIO.

- * F. Anónima (sin nombre): en JavaScript para Curryficar

- * **Curry** ficar : Parametrizar una F. Secuenc., luego ir concretándola poco a poco. Ej. abajo JavaScript con F. Anomimas. En cada Llam. la salida no es un Num. sino F., solo la concreción final da un Num. Creado por Frege, Moses Schönfinkel y Haskell C. Hist. L.. ver Progr. Fun., IEEE 754, Expon., Combin., Cat.

```
// Ej. de JavaScript con F. Anonimas
```

```
function sumar(a,b,c){  
    return a+b+c  
}  
  
function sumarCurry(a){  
    return function(b){  
        return function(c) {  
            return a + b + c  
        }  
    }  
}
```

```
const sumarCurry2 = (a) => (b) => (c) => a+b+c // otra escritura
```

```
const fa = sumarCurry(5)  
console.log("fa :" + fa)  
const fab = fa(6)  
console.log("fab :" + fab)  
const fabc= fab(9)  
console.log("fabc:_" + fabc)  
console.log("fabc2:_" + sumarCurry2(5)(6)(9))  
console.log("Normal:_" + sumar(5,6,9))
```

- * Hay P. F. React. (Event.)

- Progr. Dirigida por Event.

- Progr. Visual o Block: LabVIEW, Scratch, Simulink, MATLAB, VisualBasic, Visual Studio, VisualBasic .NET (Microsoft), Packet Tracer (CISCO), Prototipo V. (Req.) Ver V. Dat. Masiv., Vista, GUI

- Progr. Modul.: desCompo.sición a alto-nivel del programa. Ej: Progr. Concurr. es un subtipo.

- Otros: L. Concurr. (Calc. Pi) vs Secuencial (el resto, Lamb.), Progr. Estr., Pas. Mens. (Obj. Distrib. CORBA Progr. Compo.)

25.4 Otras clasificaciones de los lenguajes

25.4.1 Segundo compilación

- *Traduc.*: Tab. de Ventajas e Inconv. de *Compil.* vs *Interpr.*
 - *Compil.* (*C*, *Cpp*)
 - *Interpr.* (*Bash*, *Script*, *Python*)
 - *Maq. Virt.* (*Java JVM*).

25.4.2 Segundo concurrencia

- Tenemos
 - *Secuenc.*: la mayoría
 - *Concurr.* y *RT*: ver *ADA*, *LabVIEW Event*.

25.4.3 Segundo tipo de aplicación u objetivo

- *Turing Compl.*: *Virt.* todos lo son pero mejor *L.* según *Probl.* (elegir el adecuado según *Dis./Impl.*)
- *L.* *Propos.* o *Domin.* (*Ing.* *Softw.*):
 - General o Multipropósito o Multi *Parad. Progr.*: *Python*, *C*, *Java*, *Pascal*
 - Específico: *Internet*: *PHP*, *JavaScript*, *HTML* *BD*: *SQL* *Comercial*: *COBOL*. *Cientif.*: pocas I/O *Perif.* pero *Calc.* *Complejo* (*Matrices*, *Ec.* *Dif...*) *Fortran*, *MATLAB*, *Wolfram*, *R*. *API*: *CUDA (GPU)* *Artist.*: *Music.* (*MIDI Finale*, *XML*), *Imag.* (*P5.JS*, *JavaScript*)

Ver Tipos *Comput.*

25.5 Componentes comunes de los lenguajes

- *Atom.* : los elem. básicos o *Ud.* que se pueden *Compo.* según *Sintax.*
 - Caracteres: Letras, Dígitos, Signos, Espacios blanco/Tabulador (*Separ.*)
 - Palabras: en *LISP* ver Fig. 25.8 *Sintax.* BNF

Ver *Token*, *Instr.*, *Lin.* (*Concurr.*), *Transac.*, *Restr.*, *Atrib.*, *FN*, *Frege*, *Block*

- *Estr. Dat.* (o *Tip. Dat.*): Simples./*Comput.* (*int*, *char*). Compuesto: *Struct*: *Estr. Dat.* *C.* *I. E.. H.* (ver)
- *Token*: los 6 (*Id.*, *Separ...*)

- **Expr.** : *FBF Compo.* de *Op. eradores y Operandos (Var., Cte)* con *Separ.* (prioridad y **paréntesis=Ord. de Concurr.!**) que puede ser *Eval.*. Normalmente *Predic.* de *Log.0*. Distinguimos: a) *Aritm. (Algebr.)* $2*(3*x+pow(x, y))$, b) *Log.* (AND, OR, NOT), c) *Rel.* o Comparac. ($x < 4$) y d) *Caden.* 'ab' + 'cde'

- Statement (*Decl.*): está *Compo.* de varias E. y no se *Eval.*, se *Ejec.*
- Poder E. (*Chomsky*).
- E. Reg.: L. *Decl.* permite hacer *Busq.* en *Texto* con un *Form.* o *Sintax.* Ej: *Comando ls *.txt* o *grep Ej: scanf (TPunt.)*.

Ver E. *C, SQL, Reescr., Condic., Eval., Frege*

- *Instr.:* Ud. min. Procesable. de I/O (*scanf*), Estr. Control (*if*), Asign (=), Decl. (obj.)
- *Bibl.:* *include (C, Cpp), import (Java, Python)*

25.6 Curva de aprendizaje y popularidad

- **Popular :**
 - *Ind. TIOBE: Metr.* P. de L. basado en el nº de resultados que dan motores como *Google*.

Ver *Empresa, Estand. Facto, L. Moore, Actual., Benchmark, SDRAM, Mod. Rel.*

- Curva **Aprend.** : gráfica entre competencia y *Tmp* en un L.. Ver *Aprend. Autom.*

25.7 Lenguajes populares por orden cronológico

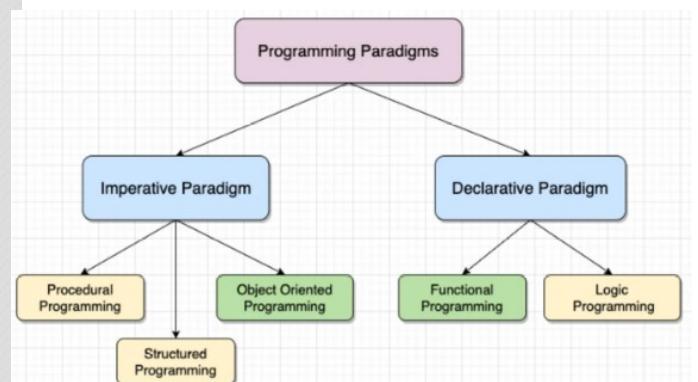
L. *Popular Actual.* según TIOBE (cada vez más *Progr. Fun.*, ver *Map.*)

- **Fortran** (Formula Translation): L. *Propos. Cient.. Anal. Numer.* El 1º L. *Nivel Alto [IBM57]* (*Hist. L.*). *ANSI77* estandariza para *Portab.*. Ver *MATLAB*.
- **LISP** (Lista Processor): *Progr. Fun.. De [McCarthy58] MIT (Turing A.). Notación Calc. Lamb.: Separ.* totalmente con Parensis, *Recurs.* y *Compo.* F.. *Tip. Dat.:* *Lista* (de F.) y *Atom.* (ver Gram. NFB Fig. 25.8). Para *SHRDLU (NLP)* Common L. tiene *POO*.
- **COBOL** (Common Business Oriented Language): L. *Propos. Comercial, Imper.. Por DARPA59.* Para Gestión y *Almac. Masiv.* de *Fich.* y *BD.* Inconv.: *Sintax.* rígida y poco *Eleg.* . Ej: Bancos *CICS*.
- Otros:
 - BASIC65 (simple para enseñar, *VisualBasic* vs *POO*).
 - PASCAL: de Wirth70 (*Modul.*) para *Eleg.*, Fuerte *Tip.*, basado en ALGOL y en que se basa MODULA (Wirth *Modul.*).
 - ADA80: *Concurr.*

- *C,Cpp* y *CSharp*
- *CSharp* : de .NET permite *Movil BDOO??*
- *Java* : *Portab.* independ. de plataforma gracias al *JVM* (es medio *Interpr.* pues la *JVM* si se *Compil.*). De Sun M. *Oracle* Gosling95.
Creado para resolver Probl. de *Punt.* de *Cpp*. Algunos lo consideran *POO* puro. Es *Interpr.* Útil en *Internet* gracias al *Applet* *Servlet* pero sobre todo en Apl. *GUI* para *Movil* y PCs como *Videojuegos* o *Formularios*.
En decadencia por Canvas *JavaScript* (ver Processing P5.JS) (*Hist. L.*). Permite *Persist.* (*Jakarta Progr. Compo.*). Sintaxis como *Cpp* pero más simple, elimina el control de L. *Nivel* bajo que lleva a error (no *Punt.* pues *Pas.* Ref. y Recol. *Basura*). Permite Multi*Hilo*. Muchas *Bibl.. Docum.* (ver Aguadulce *IDE*). Compite Oracle Java compite con .NET.
 - JDK: hasta la fecha solo 3 (8, 11 y 17) son Long-Term Support (LTS).
 - JSE

Ver *Applet*

- *Python* : muy legible (*Eleg.*) por *Separ.* indentado. De [Rossum91] (Univ. Holanda, *Hist. L.*). Multi*Parad. Progr. (Imper., POO y Progr. Fun.)*. Ecosistema (*Numpy Arrays* como *MATLAB*). Fuertemente *Tip.* (ej. *string+int*, *Cast*). Curva de **IMPORTANTE Aprend.** rápida. *Interpr.*. Tiene 4 *Estr. Dat.* famosas (*Dic.*, *Tupla*, *Lista* y *Conj.*) Ver *Aprend. Autom.*. Tiene *Generic* (*Gener.*). Ver PyGame o PyTurtles (*Videojuego*), *Google Colab*
- *Wolfram L. o Mathematica*: L. Propos. Cient. Comput. del *Conoc.* (*NLP*) “España.Fabricas/España.Pobl “derivate log(x)+x”. *NLP*
- *MATLAB* : L. *Propos. Cient.. Efic. o Filos. Matr.*: evitar *Estr. Dat.* complejas y *Lin., Ind.* o *Enumер.* todo con *Arrays* (ver *NP vs P*). Uso de *Transf. FFT*. Esto ha inspirado *numpy* de *Python* (para *GPU* y *Aprend. Autom.*). También R (Estadístico) o Scilab (su vs *Libre*). Tiene *Simul.ink* Progr. *Visual. Celdas* (*Lista* de L.). *IDE* para *Efic., Filtr., NaN*
- *Haskell* : *Progr. Fun.* (luego *Decl.*), ha influenciado a *Lean*. Ver *Curry*.
- *Lean* : para *ATP* y *Test* Caja Blanca. Basado en Calc. *Constr.* (como Coq) e influenciado por *Haskell*.
- *Scratch* : Progr. *Visual* o *Block*. Ver *App Inventor, Arduino,...*
- *JavaScript, PHP, HTML*
- *.NET* : Framework de *Microsoft* para *Win, Linux* que soporta *Cpp, CSharp, F Sharp*. Compite con *Java Oracle*, tiene *Visual Basic .NET, Visual Studio .NET, Rx.NET (Event.)* de *Meijer*.
- Otros: *Diagr.* Estados, *UML*..., Motor *Videojuego*

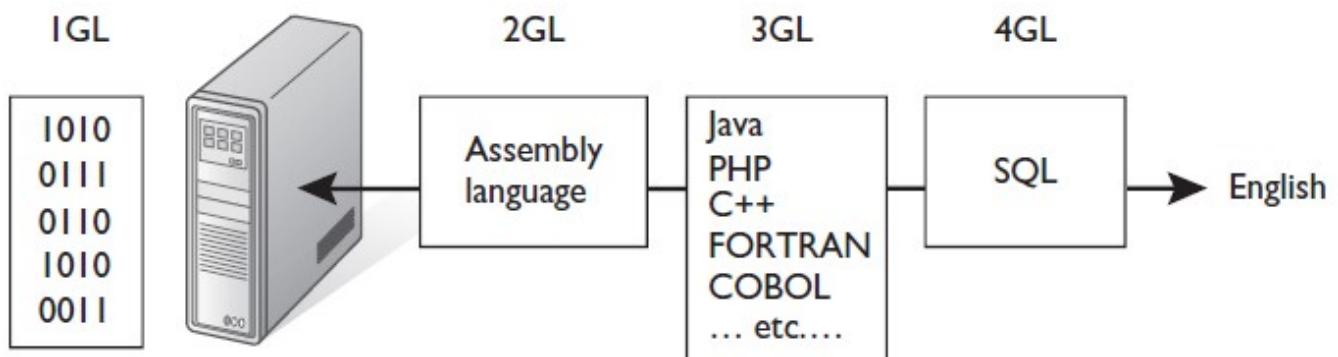


Token name	Sample token values
identifier	x , color , UP
keyword	if , while , return
separator	} , (, ;
operator	+ , < , =
literal	true , 6.02e23 , "music"
comment	/* Retrieves user data */ , // must be negative

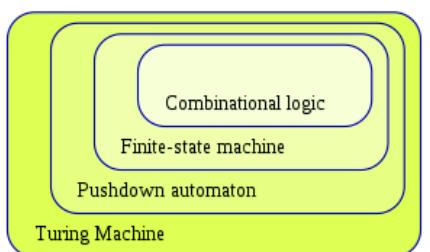
LISP syntax

```

expression ::= atom | list
atom      ::= number | symbol
number    ::= [+ -]?['0'-'9']+*
symbol    ::= ['A'-'Z' 'a'-'z'].*
list      ::= '(' expression* ')'
  
```



Automata theory



Chapter 26

+25. Programación estructurada. Estructuras básicas. Funciones y Procedimientos. (4MarPedro)

26.1 Introducción. Conclusión

- ¿Nº Instr. min. para *Turing Complet.*? Sol. Tma Progr. *Estr.*
- *Evol. Abstr. (Hist. L.), Dijkstra68 GoTo*
- Futuro: *Progr. Fun.*

26.2 Fundamentos

- *Desarr. vs Progr.*

26.3 Programación estructurada y teorema

- *Progr. Estr.* (no confundir con *Arq., Modul., Struct*): *Parad. Progr.* (de tipo *Imper.*) para mejorar la claridad (*Eleg.*) mediante el uso de:
 - a) *Subrut.*,
 - b) *Block* y
 - c) las 3 *Estr. Control* básicas (sec., if y for ver Tma abajo). Es *Unif.=Form..* Ver *Conoc., Biolog., BD, Esquema, Jerarq., Estr. Dat., Descr., Arq., Cable E. (Par Trenz.), SQL, Dis./Impl. E. Serielizar*
- *Tma de Progr. E. o de Böhm-Jacopini* (italianos): un *Progr.* Propio lo es si:
 - a) \exists_1 pto de I y \exists_1 pto de O.
 - b) \exists caminos que pasan por todas las *Instr.* (creo no sobran Instr. o no Redund.).
 - c) \forall Instr. son finitas (\nexists Bucles ∞).
- *Def.* posiblemente *Equiv. a Alg.??*

El Tma *Dem.* que cualquier *F. Turing Comput.* o (Progr. Propio) se puede *Impl.* con las 3 *Estr. Control* básicas Fig. 26.10:

- a) *Secuenc.*,
- b) *Condic.*,
- c) *Iter.* y que el *GoTo* no es necesario.

- *Hist. L.* La P. E. emergió en los 60, sobre todo tras el desaconsejo del GoTo por [Dijkstra68 Go To Statement Considered Harmful] (ver UD).
 - Harel dice que es un *Tma de Facto (Estand.)* que aprox. ya anunciarón en el 46 *Neumann* y Kleene.
 - Mills en *IBM70* tras leer a Dijkstra puso esta y el *Dis.* Top-Down de moda en la Industria por su éxito en proyectos Softw. Otros son Floyd (ver *Parad. Progr.*) o Hoare (*Ord.* Quicksort).
- Ventajas:
 - a) *Eleg.* Legible (por no *GoTo* y rupturas como *exit*).
 - b) *Efic. Ejec. Vel.*
 - c) Otras: derivadas de *Estr. Organizada*: *Depur. Vel.*, *Manten.* menor. *Impl. Vel.* (productividad en *Desarr.*) *Docum.* (los Bloq. se autoexplican).

Inconvenientes:

- Monolit. o único *Block Progr. (Arq. Kernel)*. Sol. partir en *Modul.*

26.4 Diseño Top-down

- *Dis.* Top-Down: Ventajas (*Reus., Cooper....*) y Mills (ver arriba)

26.5 Estructuras de control

- *Estr. Control* (Control Flow o *Estr. Control Flujo*): *Ejec.* conj. de *Instr.* según *Condic..*

Se pueden *Compo..*

Hay de muchos tipos: Progr. *Estr..*

- E. C. Básicas: las 3 del *Tma P. Estr.* (Fig. 26.10):
 - a) *Secuenc.*,
 - b) *Condic.* e
 - c) *Iter.*
- Otras: **Ruptura** (*GoTo*).

Otra clasificación??:

- E. C. Simples: (*for, if* de *C*)
- E. C. Dobles, Triple, Múltiple: ver abajo *if, else if, else*
- E. C. *Block*: (*if, fi* en *Bash* o indentación en *Python*) (*Interpr.*).

Es *Unif..* Ver *Pal. Reserv.*, E. C. de *C*.

26.5.1 Secuenciales

- *Secuenc.* (no *Concurr.*): un SubProgr. luego otro SubProgr, o *Block* a *Block* (*Interpr.*).

26.5.2 Condicionales

- *Condic.* (o Selectivas=*Predic.=Topos*): *Estr. Control* que *Ejec.* un *Block* según ocurra *Expr. Log.*. Hay:
 - Simples: *if*. poner *Diagr.* Fig. 26.10 de las que siguen
 - Dobles: *if-else*
 - Múltiples: *switch*, *if-if else-else*,

Ver *Restr.*, *Req.*, Ligadura (*Fisic.*). *Sincr.* C. (*Excl.*)

- *if-else*:

```
if (expr1) {  
    sentencias1;  
} else if (expr2) {  
    sent2;  
} else {  
    sent3;  
}
```

Es *Unif.*. Ver *Esper.* Activa, 4 C. *InterBloq.*, *BiEst.* C. (*CODE2*), *Busq.* *Combin.*, *Fisic.*, *Backtrackin*

- **switch**: caso de *Condic.*. *break* opcional para no seguir evaluando. *default* Fig. 30.10

26.5.3 Iterativas

- *Iter.* o Loops o Bucles o Repetitivas: *Estr. Control* que ejec. un *Bloq.* mientras se cumpla *Condic.* basada en *Testear* un Contador (*Semaforo*, Bandera, Centinela..) que *Var.* *Equiv.* a *Map.*..

Sol. más *Efic.* que *Recurs.* pero menos *Eleg..* Hay:

- *while*: de 0-N veces, ojo a ∞
- *do-while*: de 1-N veces
- *for*: se sabe n° veces

Ver *Desarr.* Agil, *Esper.* Activa,

- *for*: varini y varfin deben ser de tipo *Ord.inal*.

```
for (inicio ; condicion ; actualizacion){  
    Bloque;  
}  
}.
```

- *while* y *do-while*: ojo evitar Bucles Infinitos (*Iter. ∞*) (Fig. 30.10). OJO: evitar bucles ∞

26.5.4 Rupturas

- **GoTo** : Estr. Control de Ruptura para ir a una linea cod. Reemplaza a *F* en *Ensambl.*. No desaconsejada por Tma Progr. *Estr.* y *Dijkstra* (ver) por dar *Circ.* y **Cod. Espag.** (ver *Diagr. NS, Eleg.*). Ver *Llam. Subrut. B CODE2*.
- Otras: *break* (finaliza un for, while o switch). *continue* (pasa control a siguiente iter.). *return*

26.6 Subrutinas y Funciones

- **Main** : Progr. principal que *Llam.* a SubProgr (al terminar regresa al M.) o *F*. donde comienza y termina el *Flujo* del Progr. Ver Progr. *Modul.. Estr. C*

- **Subrut.** (no confundir con *F*. ver aquí): “*Ud. Llam.able*” de uso Frecuente con 0 (*void* vacio, O. como *Param.* I. *Pas.* por Ref.) o 1 Outputs. Se suele *Llam.* con *GoTo* (no con una *Expr.*).

Se diferencia de *F*. en que (por *Var. Global*) puede dar dif. O para mismo I, i.e.

La *F*. es una *Correspond. Unic.voco* (para mismo I. siempre mismo O)[Altenkirch19].

Según el lenguaje se nombra: *Proced.*, routine, subprogram, *F*. o method. Ver *CODE2*, *Pila*, *Recurs.*, *Punt.*, *Manej. Interr.*.

- **F**. (no confundir con *Subrut.*): I. (varios *Param.* o *Arg.* o *Var.*), O. (un valor *Unic.voco*). Hay de muchos *Tip.* Matem. *Caden.*

- *Decl.* *F*.: normalmente en Comienzo del *Fuente*, suele ser la *Cabecera*: con *Tip.I/O(Param.)+Nomb*. Para que el *Compil.* chequee la *Consist.* de *Tip.* (ver). En *C, Cpp* o *CODE2* terminan con *return*. También de *Var. Global (C)*.
- **Def.** *F*. o *Vincul.*: se hace una vez pero puede *Llam.* muchas (ver *Cl.*). Puede ir en *Bibl.* o *Modul..* a) *Cabecera*: ver *Decl..* b) **Cuerpo** o *Bloq.*: *Var. locales e Instr.* *) *ReDef.=Sobrecarg.* Ver D. *Obj. (Recurs.), Progr. Fun., Descr., typedef (Struct)*.
- *Llam.* *F*. (ver)
- Tipos: a) del Propio Lenguaje: def. en *Bibl.* (matemáticas, cadenas...) b) del Usuario: mantener estándar de calidad.

Ver *Conoc.*, *Hash*), *F. Virt.* (*Polimorf.*), *Predic.*, *Dis. F.* (*Dis. Top-Down*), *Punt.*, *Pas.. Progr. Fun.* (*Punt.*), *F. Hash*, *Ley*, *Req. F.*

26.7 Parámetros, llamadas y pasos

- **Llam.** *F*. (no confundir con *Ll. Subrut.* ni *Pas.*): lugar del Progr. *Main* donde se *Eval.* o *Pas. Param.* a *F*. Es una *Interf.* (o *Expr. Cod.* como *Decl.*) ver *Driver SO* Ver *Macro, Transpar. Main, Ambito, Pila Ll.* (*SP CODE2*), *Interop.*, *POSIX*
- **Param.** o *Arg.*: lo que ponemos a la *F*. pueden actuar como ambos *I/O* (*Pas.* por Ref.). Pueden ser *Cte*, *Var.* o *F.* (*Curry*). Distinguimos:

- RESUMEN: P. Form. (los de dentro de F.) y P. Actual. (los pasados por Ref. o Valor, ver) pero en cuanto se hace la *Llam.* los Form. toman valor de Actual.
- P. *Form.al*: *Var.* que aparece en la *Cabecera* de la *F.* y con las que *Op.era* dentro esta. (*v,N* en Fig. 27.11).
- P. *Actual.:* el especificado en la *Llam.* (del Progr. *Main*) Ej: (*ve,L* en Fig. 27.11). Se *Pas.* por Valor o Ref.

Ver *Config., Gener., Curry*

- Estrategias de *Pas.* o *Eval.* (no confundir con *Llam.* o P. *Mens.* *Kay*): forma de *Transm.* Info a una *F.* (*Expr.* o *Predic.*) (=Subst.+*Llam.*+*Proc.*+*Ver* (Wittg.)=*Consult.=Impl.*)
 - P. por Valor (Call by Value): se crea una copia de *Param.* *Actual.* que reciben los *Param.* *Form.* y los cambios hechos dentro de *F.* no afectan en el *Main.*
 - P. por Referencia (by Ref.): se P. *Direc.* Mem. (o *Punt.*) del *Param.* *Act.* que es *Compart.* por Progr. y Subprogr. Los cambios internos se reflejan en el *Main.* Para:
 - a) *Transm. Masiv.* (*Array* o *Struct* de *C*).
 - b) *Param.* actuan como ambos *I/O* (se meten y salen modif. por *Efic.*)
Ej. **v* en Fig. 27.11. Ej: *Subrut..*
 - P. por Necesidad: tipo de Memoización (Memoization ver *Efic.*), Perezosa (Lazy-ness)

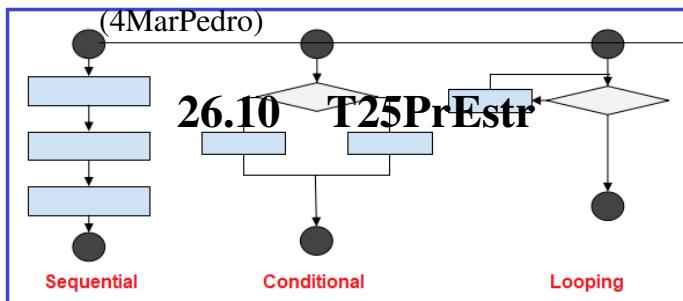
Ver Mod. *Direc., Enumer., Superpos. Cuant., Curry*

26.8 Ámbito de un identificador, variable global

- *Ambito*: *Var.* Global desaconsejada por Efecto Laterales (aumenta *Acopl.* entre *F.* y va contra *Axiom. Ortog.*)

26.9 Programación en la actualidad

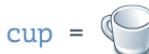
- Progr. *Estr.:* aparece en 70 y hoy sigue
- Progr. *Modul.:* si Progr. grande *Modul.*. Un *Modul.* en *C* es un *Arch.*, en *Java* se usan *Paq.* y en *WinVS Bibl.* .*ddl* de *Enl. Din.*
- *POO*: más *Abstr. Encapsul. Est.* y *Op.* en *Obj.* agrupados en *Cl.*. Los *Obj.* se Comun. con *Mens.* *Kay*. Ej: *Smalltalk, Cpp, Java, VisualBasic .NET,*
- *Actual.:* *GUI+Internet* uso de *IDE, Desarr. RAD, Bibl.* y (*WYSIWYG*).
- *Maq. Virt.* para *Portab.* como *Java* o *Visual Studio .NET*
- *L. 4GL: Decl.* y *Progr. Fun.* para *BD* como *MySQL*



pass by reference



pass by value

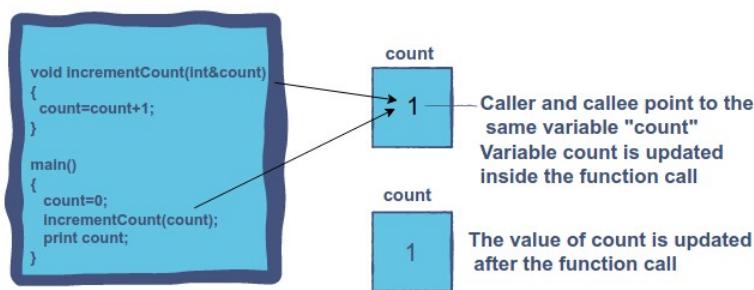
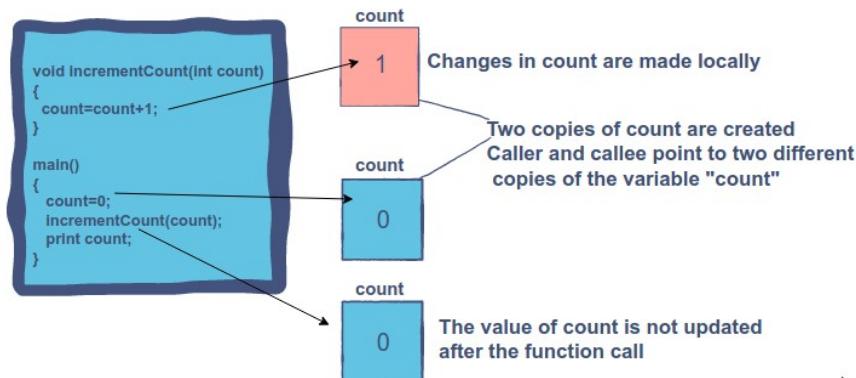


cup =

cup =

fillCup()

fillCup()



```

4 int PrintAddV (int *v, int N, int s)
5 {
6     int i;
7     if (s>=0 & s<100){
8         printf("s added\n");
9     } else if (s<0) {
10        printf("s error\n");
11    } else {
12        printf("s too big\n");
13    }
14    for (i = 0; i < N; i++) {
15        printf ("%d ", v[i]); v[i]=v[i]+s;
16    } printf("\n");
17    return 1;
18 }
19
20 int main ()
21 {
22     int L = 5, i = 1, j = 0;    float a, b;
23     int ve[] = { 4, 1, 9, 6, 7 };
24     char s1[] = "Hola ", s2[20];
25     //Vector
26     i = PrintAddV (ve, L, 10);
27     i = PrintAddV (ve, L, 0);

```

```

s added
4 1 9 6 7
s added
14 11 19 16 17
Introd. dos floats:

```

```

#include <stdio.h>
void PrintM(int *m, int nc, int nf)
{
    for (int i=0; i<nf; i++)
    {
        for (int j=0; j<nc; j++)
            printf("%d ", m[i*nc+j]);
        printf("\n");
    }
}
void main ()
{
    int m[2][3]={{1,2,3},{4,5,6}};
    PrintM(&m[0][0],2,3);
}

```

Chapter 27

26. Programación modular. Diseño de funciones. Recursividad. Librerías.

27.1 Introducción. Conclusión

- ¿Como Impl. un gran programa *Cooper*.?
- *Hist. L.*: Evol. Abstr. MODULA Wirth
- Futuro: Modularidad: aparece en *Dem. matemáticas* (en *Lean*),

27.2 Programación

- *Progr.* (Inform.)
- Conceptos: *Div. Venc.*, *Dis. Top-Down Encapsul.* (idea de Progr. *Modul.*)

27.3 Funciones y subrutinas

- *Main*: regreso tras *Llam.* F.
- *Subrut.*:
- *F*: *Param.* *Llam.*,...Local vs Global (ver desaconsejo del *Acopl.*)

27.4 Programación modular y módulo

- *Progr. Modul.* (no confundir con *Modulac.* o Modular (F. *Hash*)): *Parad. Progr.* *desCompo.sición sucesiva* (refinamiento *Dis. Top-Down*) del Progr. en Mod. (*Bibl.* o *Capa*) que contienen lo necesario para ejecutar una *F*. que *Interop.* con la de otros Mod.
 - Ej: *Pila Protoc.*, *CSS*, *Concurr.* o Arq. *SO* (donde lo contrario es Arq. *Kernel Monolit.*).

- Es *Evol.* en *Abstr.* (*Hist. L. UD.*) nace con MODULA del suizo ETH Wirth (*Turing, PASCAL*) y que iba en el PDP-11 de *DEC*
- No confundir con:
 - a) Progr. *Estr.*: descomp. en piezas de F. (a bajo nivel) mientras que M. es d. del Progr. Total (a alto nivel).
 - b) La M. da más *Eleg.* y sirve para *Probl.* más grandes.
 - c) *POO*: descomp. en obj. (a nivel de *Estr. Dat.*)
- Surge:
 - a) Progr. Principal (*Main*) y Subpragr. (o Modul.).
 - b) Subprogr. Int./Ext.
 - c) Obj. Loc./Glob.
 - d) *Param.*

Es *Unif.* por *Pila Protoc..* Ver *Conect., Tarj., RAM, ATP, Part., Fact..*

- **Modul.**: es un Progr. pero con una *Interf.* de comunicación bien definida (mediante *Metod.* de I/O). Ej.:

- *F.* o *Subrut.*: son los paradigm.
- *Block* de un Progr.: que resuelve una tarea típica:
- *Fich.*: conj. de *F.* que puede ser **Interno** (en mismo Fich. que el Principal) o **Externo** (puede estar en otro *Cod.*, ya *Compil.* independientemente y luego *Enl.*.)
- Ej: *Pila Protoc.*, Arq. MultiTier , Arq. ANSI/SPARC

Es un *Blackbox*.

27.5 Ventajas y características: Transparencia, ámbito y refactorización

Ventajas de la *Modul.*:

- *Reus.*, *Eleg.-Legibilidad. Integr.*
- Las de *Desarr.*: *Manten.*, *Manej.*abilidad
ReFact.: mejorar *Efic.* y *Eleg. Transpar.* (sin que lo note Usr)
- Las de *Dis.* Top-Down: *Cooper.* cada uno hace una parte. por *Div. Venc.*
- Alta *Cohes.* y baja *Acopl.*

27.6 Espacio de nombres vs ámbito

- *Esp. Nombres* (no confundir con *Ambito*): conj. *Unic.* de *Id.* similar al *Domin.*, forman un *Arbol.*
- **Ambito** (Scope, Alcance, no confundir con *Esp. Nombres*): lugar del *Fuente* de un *Progr.* (o *Ctx Semant.*) al que pertenece un *Id. Vincul.* (*Var.* o *F.*), i.e. desde donde es *Acces.* o *Llam.ble.* Puede ser:

- **Local:** solo Acces. dentro del F. o Subprogr.
- **Global:** desaconsejado por **Efecto Lateral** (ver *Acopl.*). Siguen una *Jerarq.* de A. que es todo el progr. salvo nombre repetido en F. y prevalece el de F. (ver *CSS*).
- *Conj.* o *Topos:* *Diagr.* de lugares (partes de *Fich.*, *Modul.*, *Dir...*) del A.

Además:

- *Op.* Resol. *Ambito:* (Std::) ver *Metod.* (*Meijer*) *Her.*
- Ventaja: hace *Transpar.* la P. *Modul.* (cambios en una parte no afecta a otras.).

Otros significados:

- SO: *Sist. Arch.*,
- R.: *DNS*, *Recurso Red* (*Servic. Dir.*), *Topol.*, *TCP/IP vs OSI*

Ver *Acopl.*

27.7 Diseño de Funciones: axiomatización ortogonal

- *Dis. F:* *Dis.* Top-Down vs Bottom-Up.
- *Axiom.atización Ortog.:* interesa alta *Cohes.* y bajo *Acopl.* Fig. 27.11, i.e. elegir la min. cantidad de F. que sirvan lo máximo *TAI*. Estas son 2 *Metr. Softw.* en Progr. *Modul.*:
 - ***Cohes.*** : ubicar en el mismo *Modul.* subF. y var. relacionadas Ej: una *Cl.* tiene alta Cohe si la clase tiene métodos similares.
 - ***Acopl.*** : *Depend.* entre *Modul.* (comparten *Ambitos*).
 - * ***Var. Ambito Global:*** están desaconsejadas si queremos reducir A. y evitar que las F. tengan *Depend.* con otras partes del Código [WikiFunction]. Además aunque haya una *Jerarq.* (ver *Ambito*) pueda dar **Efecto Lateral** (*inControlable*) si una F. modifica una var. global sin quererlo (por *Err.* por tener el mismo nombre).
 - * **Infierno de las *Depend.*** es cuando el *Graf.* es muy complejo (necesitas instalar muchas *Bibl.*).
 - *Cluster* o *Clique Graf.* en Progr. *Modul.:* “Often modules form a directed acyclic graph (DAG).... indicating that these should be a single module.”

Es *Unif.* equivale a *FN*. Ver *ATP*.

27.8 Iteratividad vs recursividad

- ***Iter.:*** más *Vel.* y *Efic.* pero menos elegante que *Recurs.*
- ***Recurs.:*** *Parad. Alg.* para resolver *Probl.* donde la solución depende de instancias más pequeñas del mismo *Probl.* (*Div. Venc.*).

Consta de 2 partes: la llamada al problema anterior (o anteriores) y los casos base (equiv. a *Dem. Induc.*).

Más eleg. que *Iter.* pero consume mas *Recurs.* de Esp. (se hace con *Llam. Pila* de *Subrut.*, ver *SP CODE2*). Distinguimos:

- R. **Directa** un módulo se autollama.
- R. **Indirecta** un mod. A llama a otro B y viceversa.
- Ej: *Fact.orial*, Fibonacci (Fig. 27.11), Hanoi o Ajedrez (*Backtracking*)
- Otros: **Rel. Recurr.**: es un subtipo de R. $x_k = f(x_{k-1})$ y que puede tener *Form.* Cerrada como Form. Binet (relac. Fibonacci con N° Áureo).
- Otros Ej: RNN: Es como una *Actual. Inform.*, *Compr.* Diferencial, Mod. *Direc.* Indirección *INode*.

Es un *Unif.* Ver *ALU* R. (*Chomsky*), *Vista* (*SQL*), *Lamb.*, T. *Church-Turing*, *Secuenc.*, *Enumer.*, *MultiLista*, *Obj.*, *Tarj.*, *Distors.*, *LPC*, *Web Embob.*, Recorrido *Arbol*, *Ord.* Quicksort

27.9 Librerías

- **Bibl.** (o Libreria mal traducido): conj. de F. Independ. (*Fuente* o *Compil.*), **Externas** (no tienen *Main*) y con *Interf.* bien definida.
Estandar de *Servic.* que el programador usa.
Permiten *Reus.* Cod..
Los SO hoy dia ofrecen Libr. En Arq. *Kernel* está entre las *API* y el *SO* (Fig. 17.11).

- *include* (*C*, *Cpp*), *import* (*Java*, *Python*), *script src* (*JavaScript*).
- *Enlazado*: en *Cabecera Fuente* se importa normal (sin Path, *Dir.*) y en *Compil.* (y *Ejec.!!*) se indica Path. Ej: *Java*: Fuente *import processing.core.PApplet;*, Compil.: *javac -cp ".:/processing-4.2/core/library/core.jar" P.java*.
- Ej: *P5.js JavaScript*, *Stdio.h* (*C*), *iostream* (*Cpp*), *numpy* (*Python*).
- Ej: B. Gráficas (*3D* o *Videojuego*) como (*Open/WebGL=Graphic Library*).
- Ej: PortAudio (*PortAudio*): *API* general *Transpar.* que llama a *APIs* según SO. Audacity la usa. Opera con *ALSA*
- Ej: *uniStd.h POSIX Fork..*

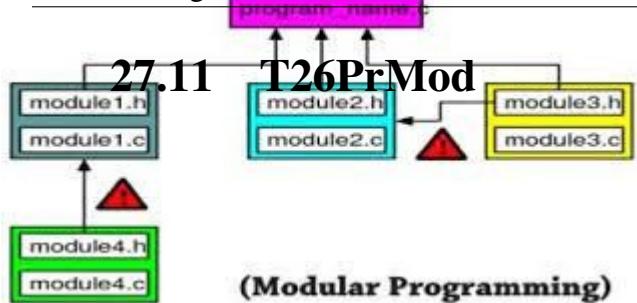
Ver Infierno *Depend.*, *Aprend.* *Autom.*, *Instal.* (*apt*), *JavaScript* (*P5.js,jQuery*)

- Estruct.: a) Llamadas a otras libr. b) Decl. glob.: var., ctes y F. c) Def. de F.
- Características: a) podemos llamar (tener *Acces.*) a cualquier F. b) Facilitan *Portab.* y el Reuso de cod. (si *Estand.*)
- Tipos según quien las crea: Libr. de Usr vs *Estand.*
- Tipos según como se crean:
 - Librerías de Cod. *Fuente*: son Texto Plano. El *Compil.* adjunta todo el Fuente al Progr. y da *Cod. Obj.* muy grande (con F. usadas o no).
 - L. *PreCompil.*: son de *Cod. Obj.*. El *Enl.* pega solo las F. necesarias. Da código grande.

- L. de *Enl. Din.* : son de *Cod. Obj.* (o *Cod. Ejec.*). Se *Carg.* en T. *Ejec.* y no en T. *Compil.* como la *Compil.*. Pueden ser invocadas por distintas Aplicaciones. Según el SO se llaman .so (Linux), .dll (Win) y .dylibs (Mac). Ej: libportaudio.so o libportaudio.dll.a. El Linux suelen estar en /libo/usr/liboLD_LIBRARY_PATH. Ver *Enl., Enl. Din. POO.*

27.10 La programación en la actualidad

- Progr. *Estr.*: la Progr. *Modul.* da cod. más corto
- Progr. *Modul.*: todos los lenguajes la permiten. En C cada archivo es un *Modul.* que pasa *Cod. Obj.* y luego se *Enl.* con paquetes (*Java*) o librerías DLL (Win)
- *POO* sigue en la linea de *Abstr.* y Reusar: *Encapsul.* el estado y Comunic. Obj. Util en *GUI, IDE* y *CASE*
- Multiplataforma: se hace con Maq. Virt. (*Java* o *Visual Studio .NET*).
- Progr. *Decl.*: 4GL (*Hist. L.*) *SQL* y *Progr. Fun.*



```

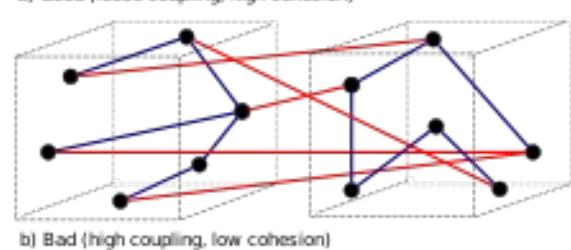
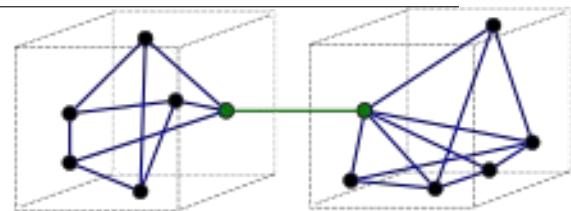
4 int PrintAddV (int *v, int N, int s)
5 {
6     int i;
7     if (s>=0 & s<100){
8         printf("s added\n");
9     } else if (s<0) {
10        printf("s error\n");
11    } else {
12        printf("s too big\n");
13    }
14    for (i = 0; i < N; i++) {
15        printf ("%d ", v[i]); v[i]=v[i]+s;
16    } printf("\n");
17    return 1;
18 }
19
20 int main ()
21 {
22     int L = 5, i = 1, j = 0;    float a, b;
23     int ve[] = { 4, 1, 9, 6, 7 };
24     char s1[] = "Hola ", s2[20];
25     //Vector
26     i = PrintAddV (ve, L, 10);
27     i = PrintAddV (ve, L, 0);

```

```

s added
4 1 9 6 7
s added
14 11 19 16 17
Introd. dos floats:

```



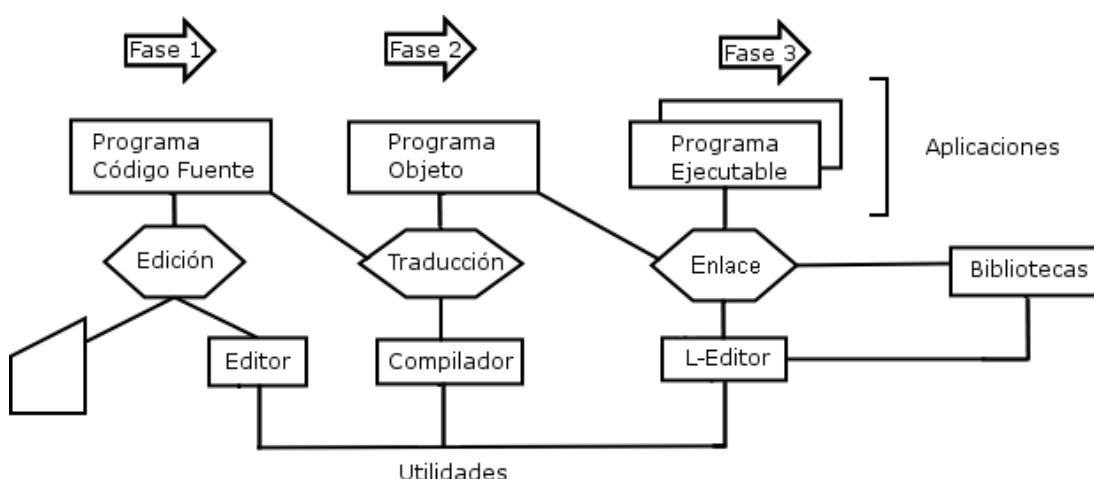
```

function fib is:
input: entero n de forma que n >= 0

```

1. si n es = 0, **return** 0
2. si n es = 1, **return** 1
3. else, **return** [fib(n-1) + fib(n-2)]

end fib



Chapter 28

*+27. Programación orientada a objetos. Objetos. Clases. Herencia. Polimorfismo. Lenguajes. (26EneElisa)

28.1 Introducción. Conclusión

- Ej. Motivador: ¿como hacer un SO *GUI*, *Videojuego* o Progr. Or. *Event.*? donde hay varios objetos que se deben dibujar en cada instante pero que cambian según el Usr los manipula. Sol. POO
- *Hist. L.*: Alan *Kay* 70 Turing A. creador de la POO, *Smalltalk*, *GUI* (ventanas en *SO*) y Pas. *Mens.* (*Polimorf.*).
- *Actual.*: *App Inventor*, *BDOO* y *Progr. Compo.*
- Futuro: *Progr. Fun.*,

28.2 Programación orientada a objetos

- *Progr. Inform.*
- *POO* : P. *Imper.* que modela la realidad como *Obj.* definidos de otros (*Dis.* Bottom-up) y que se Pasan *Mens.* (comunic. mediante una *Interf.*). Los *Obj.* *Encapsul.* los *Dat.* o *Est.* que lo caracterian) y *Op.* (def. de su comportamiento) para representar una Entidad (*Cl.* o *Tip.* *Dat.* *Abstr.* ver *Dis.* POO).
 - Ej: [Thorsten]: Type, Persona → Cuenta..
 - Es *Evol. Abstr.* (*Hist. L.*) (*Cooper.*): La POO apareció con *Smalltalk Kay* (ver) para *GUI*

Ver *Parad. Progr., BDOO, Videojuego, Progr. Compo.* (*Obj. Distrib.*)

28.3 Características: encapsulamiento y paso de mensajes

- Las características principales de la POO son:
 - *Encapsul.*
 - *Her.* y *Polimorf.* ⇒ *Pas. Mens.* (si se tiene *Acces.*)
 - Constructores y Recol. *Basura*
- *Encapsul.* (u Ocult. o Transpar.): del *Est.* i.e. de los datos miembros de un Obj. y que solo se pueden cambiar mediante las Op. permitidas para ello (*Pas. Mens.* o *Interfaz*). Se hace con *Nivel Acces. Cl.* (*public, private* ver abajo). Es un mecanismo de *Segur.* que protege los datos/op. de *Interferencias* y permite la *ReFact.* siempre que mantengamos los *Metod. publicos* (*Interf.*). *Obj.=Blackbox.* Ver *DUD.F* (*Carg. Útil*), *Tip. Dat. Abstr.*, *E. Micro, Embeb., VPN*

28.4 Clases

- *Cl.* : *Tip. Dat. Abstr.* para muchos *Obj.* (*Cl+Obj=Tip.+Var.*). Se *Def.* al principio como las *F.* con un *Id.* o Nombre.
 - *Compo.* o Miembro Cl: a) *Atrib.* (Prop. o *Estr. Dat.* que guarda el *Est.* del obj.), b) *Metod.* (Proced. o *F.* del comport. del obj. o del cambio de estado según entrada).
 - c) Constructor
 - *Nivel Acces. (Permisos)*: permiten *Encapsul.* u Ocult.
 - * *Public.:* A. por cualquier parte del programa que contiene la Cl.
 - * *Priv.:* A. solo por los miembros de la Cl (por defecto)
 - * *Protect:* Sa como sea la *Her.* Priv. nunca es A. en Cl. hijo pero Prot. si según tipo de Her. (ver Fig. 28.12)
- Puede dar *Vulner.* (Javier)
- *Sintax.:* *NombreCl, Nivel Acc (miembros de la Cl.), Nivel Acc (miembros). Lista Obj.* (Fig. 28.12)

Es una *Exten.* de *Struct (Equiv.)*, *CIDR (IP)*

28.5 Método, constructor y operador resolución de ámbito

- *Metod.* : *F.* en POO, normalmente para R/W/Proc. los Atrib. (ver Progr. *Modul.*)
 - Tipos: a) Interno: def. totalmente en la Cl.. b) Ext.: prototipo def. en cl. y desarrollado fuera (Op. Resol. *Ambito* o Doble Punto :: Fig. 28.12) c) Amigo: met. ≠ cl. pero tiene acc. a sus atrib. y met. miembros con Op. Punto (.)
 - *Llam.* *F.=Pas. Mens.* (Kay)

- **Op. Resol. Ambito** (Scope Resolution Operator): dos puntos doble ::, para *Def. Metod.* grandes fuera de la clase Fig. 28.12. También para *Her.??*. NOTA: *sdt::* se puede quitar si usamos *Bibl. #include <iostream>; using namespace Std;*. Según Erik Meijer está relacionado con Obj. *Expon.* y *Correspond.* CHL.
- **Constr.**: es un tipo de *Metod.* que inicializa los Atrib. de un *Obj.*
 - a) Por defecto: no requiere *Param.* para inicializar las variables miembros. b) De oficio: si no tiene por defecto el *Compil.* crea uno por defecto. c) De copia: cuando un obj. se crea de otro obj.
 - **Destructor**: elimina (*Libre*) el obj. del sistema. Llevan tilde delante. No necesario por Recol. *Basura* salvo en *Cpp*

```
~Animal(void) { delete [] data_; }
```
 - Recol. *Basura* : el “entorno de obj.” destruye obj. que se han quedado sin ref. *Libre* memor. (*Gest. Mem., Persist.*). Es *Unif.* por *Trim* y parte de *Gest. Mem..* Ver *Asign.* en *Sist. Arch., Free (Dinam.)*.

28.6 Objetos e instancias

- **Instanc.**: Ejemplificar o *Impl.*: de un mismo elemento *Abstr.* se pueden crear (por el SO, usr o programador) distintas Implement. o *Ejec..* Ej: (*Cl. → Obj., Alg. → Progr. → Proc., Fich. → Enl. Duro...*). Por ello *Id.* (*PID (Proc.), FID (INode), GID-UID (Usr), Arch. Dispos. ID...* Es un “se dice” Wittg.. Es *Unif.*). Ver *Tupla (BD)*,
- **Obj. Instanc.** de una *Cl.* (que es *Abstr.* o *Tip.* o caract. generales de muchos Obj.) (Fig. 28.12). Se crea como Tip. *NombreObj.* Consigue *Recurs.* gracias a Obj. que toma como Input otro Obj. (Ej: Def. *SPN*). Ej. *Socket* para *Comunic. Robot.*
 - Reglas de *Instanc.:* a) Inutilidad: las *Cl.* no se usan solo sus *Obj.* o instancias. b) ∞: de una Cl. podemos construir tantos elem. como queramos. c) Compartición de met.: 2 instancias comparten los mismos met. pero tienen dist. atrib.

Es un *Unif.* Ver *Servic. Dir., Entidad, Obj. Distrib. (Progr. Compo.)*

- Otros *Obj.:* Posee:
 - Propiedades o estado del obj.
 - Comportamiento: respuesta del obj. frente a otros. *Obj.=Blackbox* de la que solo se conoce como interactua con In/Outputs.
 - Referencia: *Id.* a un obj. y es indep. de su estado (como la *Cl.*)
 - Pas. *Mens.:* con lo que se comunican entre obj (*Kay*).

28.7 Herencia y diagrama UML

- **Her.** : mecanismo que permite obtener (creo solo ampliar, no reducir) una *Cl.* de otra. La *Cl.* hijo o derivada añade nuevos (o redefine los) miembros la clase padre o base.

ClDer = MiembrHered + MiembrProp. Ej: *Titulo* → *TituloCentrado* Ej: ver *Polimorf.* Permite *Reus.* evitando el *reDis.*, *Depur.* (verificación, simplif. *Eleg.*) de la parte ya *Impl.ementada.*

- *Sintax.*: def. como siempre la nueva clase pero añadir al nombre 2 puntos + *public* NombreClasePadre Fig. 28.12.
- *Diagr.* UML: abajo.

Es *Unif.* (ver tipos de H. *Exten.-Reduc.*)

- *Diagr.* UML: como el *Model.* *ER* nace del *Anal.* de *Req.* o *Semant.* (lectura) (ver *Dis.* POO.) *Jerarq.* de H. +/-# (*public*, *private*, *protected*), rombo (agregación), flecha (Her, sentido *SPN*) Fig. 28.12. *Estand.* por OMG, para Her. POO y Patr. Dis.
 - **Probl. yo-yo:** es un AntiPatr. si Diagr. UML demasiado complejo (no *Eleg.*)
- Tipos de H.
 - H. simple (único padre ej: coche).
 - H. Múltiple (más de un parente. ej. v. anfibio)
 - H. de *Impl.ementación*: los met. no se escriben (son heredados)
 - H. de *Interf.* (en Java) o **Cl. Abstr.** (en C++): Cl. sin Instancias para crear derivadas. ReaGrup.a métodos comunes. Ej: Cl. Vehículo Fig. 28.12. No hay Impl. a nivel de clase solo se Her. la interfaz.
 - *Exten.* (añadir) o *Reduc.-Especializar* (quitar, *Equiv.* o *Unif.*). *Struct* caso simple.
- *Acces.* a los miembros según *Her.:* como *Cl.* y *Tab.* Fig. 28.12 a) *Priv.:* se hace *NoAcc.* b) *Protected:* se hace *protected* salvo privado a privado. c) *Public.:* lo deja como estaba. Forma parte del *Encapsul..*

28.8 Polimorfismo. Función virtual

- **Polimorf.** : responder al mismo *Mens.* distintos *Tip.* de *Obj.* *Id.* con el mismo nombre a F. distintas. Permite Pas. *Mens.* *Sintax.* iguales a *Obj.* de distinto *Tip.* (que es un arg. de I.??, no confundir con *Tip.* Depend.). Ej: sumar vs concatenar: *Add(2,3)*, *Add('hola','Juan')*, *Add(queue1,queue2)* Ej: tamaño vector vs grafo *len(vector)* vs *len(camino)* (grafo). Ver *Sobrecarg.* (abajo en tipos).
- Pas. *Mens.* entre *Obj.* o *Filos.* *Kay* o *Comunic.* entre *Obj.* o *Interf.* o *Llam.* *Metod.* (si se tiene *Acces.* es la forma de nombrar la *Llam.* a *Metod.* en POO. *Kay* dice que debemos enfatizar el PM sobre los *Obj.* (*Filos.* T. *Cat.*). En C para lanzar F. debemos *Llam.* F. por su nombre, en POO basta mediante una Var. Mem. *Compart..* Provoca *Enl. Din. POO.*
 - Ej: Dados 3 Obj. (*Cuadrado*, *Rombo*, *Triang.*) *Instanc.* (o *Her.*) de Clas. *Forma* con Met. *AreaSuma* =. Para conocer el área de cada uno “envio mensaje a *Forma* con *Interf.* común *Area*” y estos “responden” con sus áreas.
 - Ej: Cl. *Pez* y Ave *Her.* de *Animal.* Un 3er Obj. envia el mensaje *Mover* por medio de un Var. de Ref. (Mem. *Compart.*) preestablecida en *Animal.*

- *Enl. Din. POO* (Dynamic Dispatch) (no confundir con *Enl. Din.*): Decidir en T. *Ejec.* (y no en T. *Compil.*) que *Impl.*ementación de un *Polimorf.* usar (*Vincul.*). Muy típico de la *POO* donde el *Tip.* de *Obj.* *Polimorf.* no es conocido hasta el T. *Ejec.*

28.9 Polimorfismos: Sobrecarga de operadores

- *Sobrecarg. Op.* (Op. Overloading, SobreCarg.): es un tipo de *Polimorf.* Ad-Hoc que reDef. un Op. de *Cpp* para que actuen sobre obj. de una nueva forma. Ej: *Add* (*Polimorf.*), S. de Op. *Asign.* (= o *Equiv.* para Constructor). Ej: *CSS*. Ver *Congest.*,
- *Sobrecarg.* de *Metod.*: el compilador sabe cual *F.* usar por el n° de argumentos o su tipo.
- *Polimorf.* en *Her.*: mediante F. *Virt.* (**Overriding**)
 - *F. Virt.*: es una *F.* que al ser *Decl.* *F. virtual* permite def. más tarde en en alguna de sus SubCl. *Her.*. Tras varias herencias y polimorfismos, definir la *F.* válida a usar para T. *Ejec.* (no en T. *Compil.*). Ej. en Fig. 28.12 Eat es la *F. Virtual* . Usa **Met. Sobreescr.** (Overriding): poder redefinir (modificar) un método hijo que ha sido ya heredado de una clase padre.

28.10 Lenguajes orientados a objetos

- Clasific. de *L. POO*:
 - L. puros: como *Smalltalk* y Eiffel.
 - L. MultiParad. Progr.: *Java*, *Cpp*
- *Smalltalk* : creado por [Kay70s] (*Polimorf.* 1er L. junto a *Simul.*a de *POO Hist.* L., Kay trabaja en *GUI*. Sintaxis simple. Solo usa obj (las Cl. también son obj.) ⇒ no usa tipos predefinidos. Todas los atrib. y met. son *Priv.*. Tiene una especie de SO propio (*Shell*) pues es *Interpr.*.
- Eiffel (lib. memoria, escrito en C es *Portab.*) y Autor ():
- *Cpp* : deriva de *C. L. Propos.* General, *Imper.*, *POO* (*Her.* multiples..) y algo *Progr. Fun.*. *Compil.*. *Bibl.* estandar de Cl. . Ej: *SO* (Win, MacOS, Linux). Chrome, Mozilla, Facebook, Web-Amazon, Office, Acrobat, y *Videojuegos* como FIFA, Fortnite. Tiene Templates (*Gener.*) (ver). Ver *CSharp*
- *Java*, *Python*, *Net* (indep. de plataforma como Java, soporta C# y *Visual Basic Net*)

28.11 La programación en la actualidad

- Actualmente por las *GUI*, Internet y Multiplataforma
- *IDE* con herramientas *RAD* (*Desarr.*) y *CASE* y *Bibl.* estandar.
- Hoy casi todos los nuevos lenguajes permiten *POO* Junto a Maq. *Virtual* permiten Multipl. e Internet. Ej: *Java* o *.NET*

- POO está en SO, GUI y BD: SQL para tratar datos (L. No *Proced.*)
- Progr. Orientada a Aspectos (POA): func. transversal (repetir en distintos módulos)
- *BDOO*: Bases de Datos Orientadas a Objectos

28.12 T27POO

Clase nom_clase:

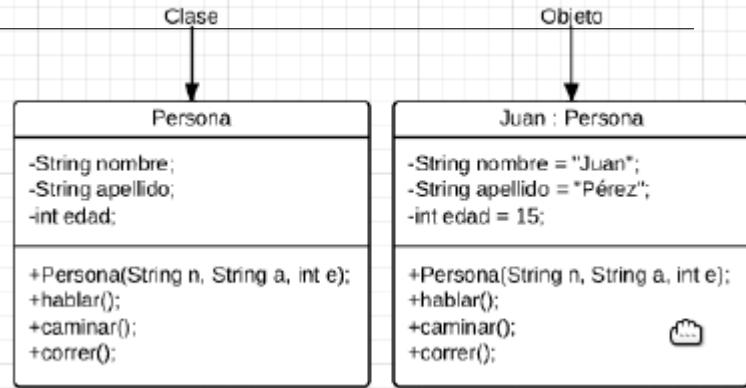
Nivel de acceso:

Miembros de la clase:

Nivel de acceso:

Miembros de la clase:

Lista de objetos:



Ámbito Herencia	CD (*) H. Pública	CD H. Protegida	CD H. privada
Visibilidad en clase base			
Private	No direct. accesible	No direct. accesible	No direct. accesible
Protected	Protected	Protected	Private
Public	Public	Protected	Private

(*) CD: Clase Derivada.

```

#include <iostream>
using namespace std;

class Rectangulo{
protected:
    int largo, ancho;
public:
    Rectangulo() {largo=2; ancho=3;}
    Rectangulo(int a, int b) {largo=a; ancho=b;}
    int area();
    int perimetro();
};

int Rectangulo::area(){ return (largo*ancho);}
int Rectangulo::perimetro(){ return ((largo+ancho)*2);}

int main()
{
    Rectangulo Ob;
    cout<<Ob.area()<<'\n';
    return 0;
}

```

```

class Animal {
public:
    // Intentionally not virtual:
    void Move() {
        std::cout << "This animal moves in some way" << std::endl;
    }
    virtual void Eat() = 0;
};

// The class "Animal" may possess a definition for Eat if desired.
class Llama : public Animal {
public:
    // The non virtual function Move is inherited but not overridden.
    void Eat() override {
        std::cout << "Llamas eat grass!" << std::endl;
    }
};

class Cuadrado:public Rectangulo{
public:
    Cuadrado(int lado): Rectangulo(lado, lado){}
};

int main()
{
    Cuadrado Ob(8);
    cout<<Ob.area()<<'\n';
    return 0;
}

```


Chapter 29

*+28. Programación en tiempo real. Interrupciones. Sincronización y comunicación entre tareas. Lenguajes. (17DicFrancis)

29.1 Introducción

- Ej-motivador: ¿como lanzar varios Proc. de un cómputo *Concurr.* y asegurarnos que su ejecuc. siempre dura igual? Sol. SO de RT. Ej: Marte LiveStream o *TeleMedicina*. Ej: ABS coche, *Satelite GPS*. Ej: *DSP cancelador de Ruido*.
- Hist.: Stallman *POSIX*, Layland73 Metr. *Plan. RT*
- Futuro: *Red de RT (IoT, TeleMusic./Medicina..., LIGO Interferometria??)*.

29.2 Definición de tiempo real: computación, SO, L, sistema

- Comput. *RT* (Real Time, *Tmp Real*): garantiza al *Usr* al menos (a):
 - (a) Previsibilidad: $t(\text{Code}) < t_{max}$, *T. Resp. Ejec.* de una misma Sec. *Cod.* tiene un máx., no solo importa respuesta *Log.* sino cuando.
 - (b) *Cambio Ctx* < t_{max} : en *SO TR*. También el t. de los otros 3 de MicroArq. *Kernel* (*Plan.*, *Mem.*, *Virt.* e *IPC*): si un Proc. *Interr.* con *Prior. Crit.* debe atenderse inmediatamente (aunque haya *Servic.*). Para *I/O* (ver abajo).
 - (c) *Robust./Segur.*: a *Fall.* bajo *Carg.* Proc. extremas. Basados en *Redund.* (más Código o Hardw. del normal). Ej: *Manej. Excep.* si no da tiempo. En *SO*.
 - (d) *Concurr.:* *Tma* todo Sist. RT es C. *Virt.!*, pues puede Ejec. Proc. *Paral.* en un t. finito. Suele venir en los L. o SO de RT (*IPC*)
 - (e) *Control Tmp*: sist. React. activado (*Trigger*) a) por t. (abrir puertas a 9:00, period. *Comando cron*) o por b) *Event.* (*App Inventor near-RT* abajo). Ej: Cl. *Clock* de medir t. *Arduino wait(500ms)*.

- (f) Alta F. *Muestr.* en *I/O*: por *Buffer* y *Recepc.* no *Bloq.* (*Esper.*). Ej: **DSP Audio cancelador Ruido**. Ej: Proc. no se detiene como *DMA*.
- (g) *Sist. Embeb.* y *Efic.*: en uso de *Recurs.. Cod.* fácil *Manten.* y *Eleg.* (*Modul.* y *Eleg.* ver *Monit. Semaforo*).

Es *Unif.* como *QoS* (garantizar). Ver *RTP*, *QoS*, *Latencia*, *Autom.*, *Event.*, *Plan.* Proc. en RT, *L. ADA*, *Conect.* Firewire, *Monit.*

29.3 Sistemas de RT, clasificación y aplicaciones: DSP y embebidos

- *Sist. RT*: Fig. 29.13 a) *I/O*: *Sens.* y *Actuad..* b) *Comput.-Progr.* RT c) *Visual-Monit.* puede ser *Tele* (*Actual. Progr.*). d) *BD*: *Almac.* *I/O* para *Manten.* y mejoras
- Clasif. *Sist. RT* según (a) *T. Resp.:*
 - Hard (*Crit.*): ej. *TeleMedicina*, marcapasos, freno **ABS** (Anti-lock Braking System), nuclear. Ver *Prior.* Critica (*Plan.* RT abajo). Con *pthread-attr-setshenparm()* das *Prior. Crit.* un *Proc.*
 - Soft (*ACritic.*): si tarda más se pierde dinero pero no vidas. Ej: *Monit. Bolsa Valores (Transac. Alta Frec.)*, Reservas vuelos.
 - Firm (Estricto): mucho retardo disminuye el *QoS*. Ej: *Multimedia*. Ej: SO generales
 - Near (Casi): ej: *Marte LiveStream*, *GPS Satelite*, un *Stream (RTP)* se ve con *Latencia* (retardo,delayed) pero en RT. Ver *React.* (*Event.*).
- Aplicaciones: Arq.: *Embeb.*, *Control Robot* fábricas (*Temp.*, presión gas, montaje.... *Calidad* y *Vel.* de producción), VA (coches), BD: *Monit. Est.* (pacientes médicos (cardiograma), aeropuertos, osciloscopio...) SO: *Multimedia* y *Videojuego* (Canvas JavaScript) sin atranques. Red: de 3 anteriores: *Satelite GPS near-RT*, *IoT (Internet no, parcheo RTP)*, *Monit. (TeleControl)* y *Multimedia (Streaming)*

29.4 Interrupción

- **Interr.** : *Petic.* a *CPU* que cambia *Flujo* normal del *Progr.* (si es posible según *Prior.*) para que se ejecute una *Subrut.* (o *Manej.* de I.). Normalmente lanzada por *Perif.*. La Fig. 29.13 distingue 3 I.: a) Hardw. (IRQ) b) Softw. (*Signal POSIX*). c) Excepç. (*Manej.*).
 - **IRQ** (I. Request, *Petic.* de I., no confundir con *Signal* del *IPC* abajo ni *IRP (I/O Win)*): I. Hardw. lanzada a CPU por una señal proveniente por un *Perif.*. Se *Manej.* con *IVT (Tabla Vect. de Interr.)* como *ITV* asocia *Petic.* *Interr.* F. *Manej.* en *Micros x86*. Ojo en *CODE2 ≠ Instr.* I. el *Cod. Ensambl.* debe tener previsto *Manej.*
 - Es *Persist.:* *Proc. → Kernel – desaloja → Interr. → Proc..* Similar a *OrCo:* -) *Dispos.* envia *Interr.* al *Manej.* de I. (MI) (es *ASincr.*, llega en mitad de *Reloj*). -) *Kernel* envia *Signal* de I. a CPU y para *Flujo*. -) *Cambio Ctx:* *Kernel* o *CPU* permite terminar *Instr.* actual y guarda *Est.* de sus *Registr.* (*PC, AC, IR..*). -) Ejecuta *Manej.* de I. -) Vuelve al *Flujo*.

- En *Masc.* I.: forma más simple Hardw. de conseguir *Excl. Mut.*
- *Filos.*: de lo mejor es que **Prod.** avise al **Consumidor**. Ej: Controlador DMA lanza I. al terminar la Transfer. Ej: *I/O ASincr.* (abajo).

Es *Unif.*. Ver *Prod.*, *DMA*, *PIO*

- *I/O ASincr.*: sol. basada en *Buffer+Interr.* para evitar consumo Recursos de *Esper.* Activa. Ej: Recepc. No Bloq. (abajo), IIO y DMA, Circ. Secuenc. ASincr, Biolog.
 - Recepc. No *Bloq.* ueante: es un tipo de *I/O ASincr.* (*Esper.*) que usa *Buffer*. Si el *Prod.* que consume la F. no está listo, La F. se pasa a Bloq. hasta que esté listo (tras unos cuantos ticks del SO). Da Plan. Promedio (arriba) en SO Normales. La R. No B. se indica con una *Etiq.* Ej: *fcntl(tube[0],NONBLOCK)* abajo.
- **Manej.** de Excepciones (Exception Handling) (o *Interr.*): *F.* o *Subrut.* que se ejecuta cuando ocurre un *Err.* inesperado en la *Ejec.* (*NaN*, Div. por 0,...). Dan *Robust.* al Progr. Permite *Comunic.* *Turing Complet.* (*Concurr.*).
 - *SigHandler()*: C.
 - *try-catch*: Java (abajo).

Ver *Mem.* *Virt.*, *Hilo*, Teclado (*Perif.*), *NoSQL*, Ventajas *Modul..* *CODE2 Overflow, Bloq..* Gest. *I/O*.

29.5 Concurrencia

- *Concurr.*: (*Fork+Sinincr.*) **Excl. Mut.** y **Sincr. Condic.**
- Mem. *Compart.* (*Semaforo, Monit.*). *Pas. Mens.* (*Sinincr.*)

29.6 Sistemas operativos de tiempo real

- *SO RT*: *Control* total, *IPC*, *Robust.* a Fall, *Concurr.* *IPC*, F. *Muestr.*
 - Ej: *QNX*, *LynxOS*, *RTLinux* Ej: *Win10 IoT*, *Azure RTOS (Nube)*
 - Suelen ir en *Embeb.* y *PLC (Autom.)*
- **QNX** : SO de *RT* gracias a su *microArq.* *Kernel* que cumple *POSIX*. Es además **SO Distrib.** Permite Obj. Distrib. (*Progr. Compo.*) (*Polimorf.*).

29.7 SO: Planificabilidad en tiempo real

- *Anal. Plan. RT*: sea un *Proc.* Emisor (o Productor *TAP*) de *Paq.* periódico T_{max} y otro *Proc.* Receptor, consiste en comprobar que $\sum_i t(F_i) < T_{max}$ i.e. que la suma de t. de las F. que *Compo.* un ciclo de E. es pequeña i.e. que se cumplen los plazos (deadline). En *SO Normal* se mide el promedio y se usa Recepc. No *Bloq.* (*Esper.*).

- *Test de Plan.*: *Metr.* de Uso de *CPU*: $U = \sum_i^n \frac{C_i}{T_i} \in [0, 1]$ donde 1 es *Opt.* y donde n Proc. periódicos con período T_i y t. Ejec. CPU C_i (con *Prior.* Estat. como RMS, pero en EDF no??).
- *Alg.* **RMS** (Rate Monotonic Scheduling o Plan. de Tasa Monótona, recordar con Root Mean Square no con SMR FP): de *Prior.* Est.ática que asign. *Prior.* mayor a los Job que duran menos (como **SJF**) y los Proc. no *Compart. Recurso* (no *Semaforos..*). Layland73 *Dem.* que estos Sist. son Plan. si $U = < \ln(2) = 0.69$ para n grande *Hist. SO*. Vemos que no es *Opt.* en uso comparado con EDF (Prior. Dinam.).
- *Alg.* Plan. *Prior. Dinam.*: al contrario que RMS la Prior. cambia según Ejec. Como **Cola Multinivel**. usado en *SO* como *Win.* Ej: **Earliest Deadline First** (EDF) que da un $U = 1$ *Opt.*
- *Alg.* Plan. *Cooper.* o No Aprop.: Round Robin (mejor esfuerzo) y *Multitar. Cooper.* (en *Embeb.*)
- IDEA: Compromiso: *Opt.* uso de *Recursos* ($U = 1$) vs *Control Tmp* (Recepc. *Bloq.*).

29.8 SO: Herramientas de diseño y análisis

- *Herram. Dis. y Anal.:*
 - Redes Petri (Calc. *Pi*)
 - *Test Caja Blanca*: para Sist. *Segur.* (ver *ATP*)
 - *Anal.* de *Carg.* de Recuros: *CPU*, *MP Times*, *Comando top*
 - *Anal.* t.: aiT (NASA, AIRBUS),
 - *Anal.* Plan.: RTDruid (integr. con *IDE Eclipse*),
 - *Simul.* Plan.: RTSim (*Cpp*) y TrueTime (sist. *Distrib.*).

29.9 Programación: estándares

- **POSIX** (Portable Operating System *Interf.*): conj. de *Llam.* o *APIs Estand.* para *Concurr.* (*IPC*). Tiene una *Exten.* para *RT*.
 - *Signal del IPC*: de forma similar a las IRQ (*Interr. Hardw.*) estas son las “*Interr.*” Softw. que usa el *IPC* para Comunic. Se *Transm.* mediante *Kernel* (con *Comando POSIX*) y *Manej.* por *Proc.*, mientras que I. es *Transm.* mediante *CPU* y *Manej.* por *Kernel*. Ej: *SIGSEGV* (de *Fall. Segment.*, ver *Null Hoare*), *SIGINT* (*Interr. Ctrl+c*), *SIGUSR1* (de *Usr*)
 - *Bibl.* con F. para *Descr. Arch.*: *uniStd.h* (como el *Stdio.h*, abajo): *Comandos* (System Call) *Fork*, *Wait* (*Esper.*), *write* y *Pipe*, y creo *Signal.* B. *fcntl.h* (ver abajo).
 - *Hist. SO*: *Petic.* del *IEEE* a Stallman para mantener *Interop./Portab.* entre SOs *Unix/Linux* (*QNX* lo usa).

Ver PBS (*Grid*). *Cluster. Descr. Arch.* (3 n°s, *INode*), Los 7 Fich..

- Otros: OSEK-VDX/AUTOSAR (Automovilistica), ARINC-APEX (aviación, Boeing777), MICRO-ITRON (*Embeb.*).

29.10 Programación: concurrente fork y wait

- **Fork** (Bifurcación, no confundir con *Hilo*): *Crea Y Ve Si OK* como *Fich.. Comando POSIX* que crea de un *Proc.* padre otros P. hijos. (2º Plano, *Background*). No *Her.* el *PCB* como los *Hilos*. (un P. tiene varios H.). Suele ir con *Esper.* (*wait*). Ver *Daemon*, Bomba Fork (*Ataq.* DOS), *Vers..*
- **Esper.** (Wait): *Comando POSIX* que suele ir con *Fork* Fig. 29.13. Ojo a los Proc. **Zombies**.
 - E. Activa (o Spinning o Busy Wait o *I/O Sincr.*): chequear continuamente (while-*Iter.*) hasta que se cumpla una *Condic.* como un *Semaforo while(Condic.)*. Defecto: consume *Recursos* como *Energ..*
Lo contrario es la *Interr.* y la *I/O ASincr.* (es mejor que el **Productor avise al Consumidor**). Ej: *Instr.* TSL (*Excl.*), PIO (*TDMA*), SMP (*Mem. Compart.*), *Recepc. Bloq.* (*RT*), *Walkie Talkie-Movil*.
 - **Polling** (o Sondeo, no confundir con *Spooling*): un tipo de *Esper.* Activa un poco más Efic. Cada cierto t. la CPU sondea los *Perif.* para ver si hay algo para él pero esto consume mucha CPU, alternativa mejor: *Interr. ASincr.* Fig. 29.13. Rel. con *Monit.*

Es *Unif..* Ver *Embeb., Perif.. Alg. Peterson. Comunic. OrCo, Petic.. E. Acotada (Inanic.)*

29.11 Programación concurrente: ejemplos en C y Java

- *L. RT:* deben permitir (a), (b),...y *Plan.* Acotada
 - *Ensambl.:* calcula bien el t. de *Ejec.* de cada *Instr.* por su cercanía al Hardw. pero poco *Portab..* Ej: *Arduino wait*
 - Prop. General: usan *Servic. IPC* del SO como *C, Cpp o Python*
 - Especif. de RT: sin *Servic.* del SO, son los de *oConcurr.* como *JavaRTS (JVM* creo no aunque le des *Prior. Critica*), *ADA* o *LabVIEW u Occam*
- *C: Fork+Pipe (Tube), Manej.*
- *Java: extend Threads y for+Th.run() Th.start(). wait() +notify() +notifyAll()*
- Ej: En *cclient.c* poner *signal(SIGUSR1, SigHandler);* y cuando el *cfather.c* lanza la S. *POSIX – SIGUSR1 kill(pidchild, SIGUSR1);*. Otras S. POSIX de *<signal.h>*: : *SIGINT* (creada al Puls. en Shell Ctr-C).
- Ej: *UDP:*

cfather.c:

```
#include <uni\hyl\Std.h> //POSIX fork, wait, ...

int pidchild, tube[2];
pipe(tube);
fcntl(tube[0],NONBLOCK); // hace vector tube un pipe No Bloq.
...
if(pidchild=fork()==0){ // crea Proc. hijo en 2 plano y ve si OK
    cchildUDP(tube,IP);
}
...
write(tube[1],"UDPF"); // manda mensaje UDPF al hijo
kill(pidchild,SIGUSR1); // manda Signal al hijo
while(wait(&status)!=pidchild); // espera cierre hijo
close(tube);
```

cchildUDP.c:

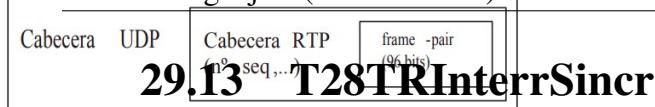
```
#include <uni\hy\{Std }.h> //POSIX fork , wait ,...
#include <signal.h> //establece Manej.

int gtube[2]; //var. global compartida (pipe)
int gstop=0; //var. global
...
void SigHandler(int sig){ //Manej. si father manda SIGUSR1
    read(gtube[0],cad); //lee non blocking mode
    if(strcmp(cad,"UDPF") == 0) gstop=1;
}
...
void cchildUDP(int *tube, char *serverIP){
    gtube=tube;
    signal(SIGUSR1,SigHandler); //comunic. Turing completa
                                //por tube
    while(gstop==0){ //lee audio y lo envia UDP
        ReadAduioBuffer(framebuffer);
        SendUDP(framebuffer);
        gstop=VAD(framebuffer);
    }
}
```

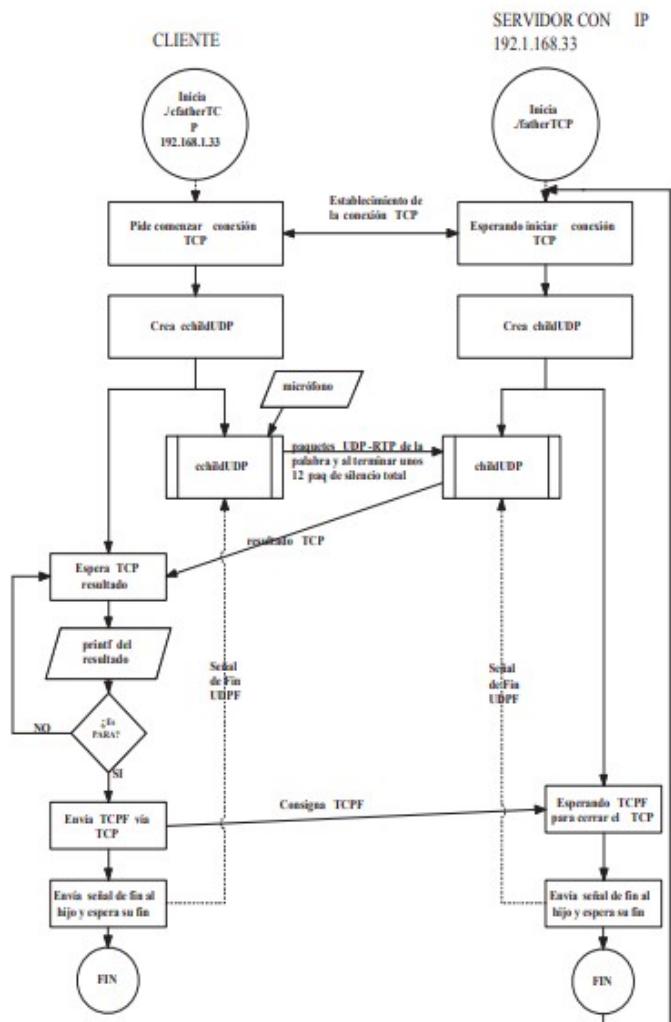
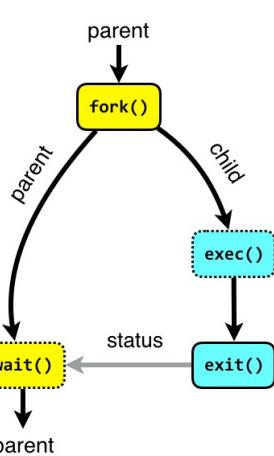
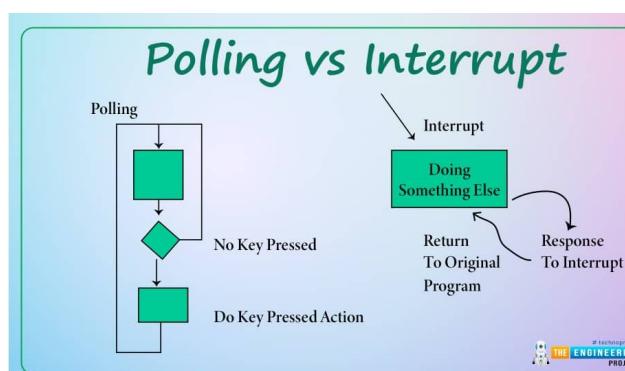
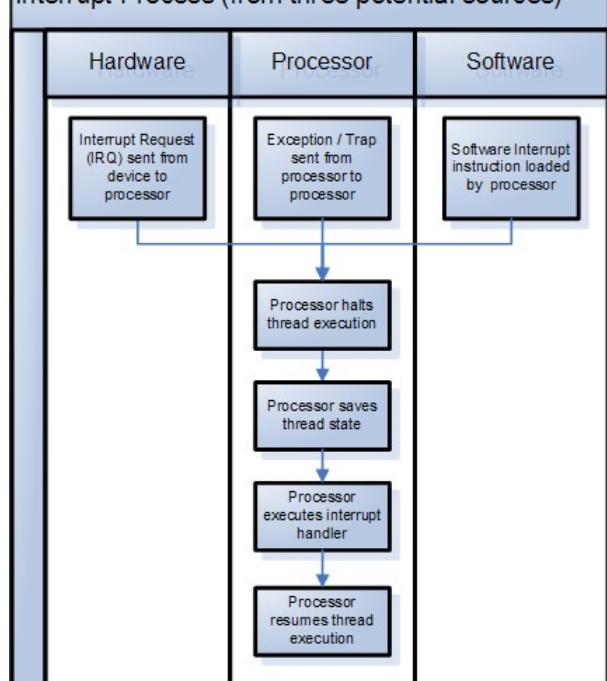
29.12 Programación: orientada a eventos

- **Event.** (tipo de *Interr.* o *Trigger*): *Parad. Progr.* para *GUI* en *RT* o *Embeb.* que espera a lanzar una *F.* hasta que se pulsa ratón o teclado. Ej: *JavaScript (mouseX/Y)* o *App Inventor Movil.* Permite *Concurr..* Con *Interr.+Esper.* Activa. Ej: Impress on demand (cuento animado)
 - Progr. **React.** (no confundir con React *JavaScript*): en tradicional $a = 2; b = 3; c = a + b; print(c)$ si cambio $a = 3$ (*Event.*) no cambia lo que se imprime, en reactiva se usa el *Op.* especial $cDolar = a + b$ que si lo cambia en *near-RT*. Ver *ReactiveX* (abajo), R. *RT*.
 - ReactiveX (no confundir con *DirectX (Render.)* ni *ActiveX (Progr. Compo.)*): L. para *Impl.* P. *React.* (ver arriba). Creado originalmente por Erik *Meijer* (ver *Expon., Cat., Resol. Ambito, Correspond., Correspond., Patr.*!! para .NET Framework y *Exten.* luego a *JavaScript (RxJS)* y otros (RxJava, Rx.NET, RxPy and RxSwift)). Los disparos (*Trigger*) son por cambios de datos y no por *Proc.* como en E.???. Se hace con *Progr. Fun..* Ver Mod. *Vista-Control*.

Ver *Monit., Intr. (IPS)*



Interrupt Process (from three potential sources)



Chapter 30

+31. Lenguaje C: Características generales. Elementos del lenguaje. Estructura de un programa. Funciones de librería y usuario. Entorno de compilación. Herramientas para la elaboración y depuración de programas en lenguaje C. (9MarFuen)

30.1 Introducción. Conclusión. Lenguaje C

- *C* : lenguaje con historia pero *Actual. Popular* (de los 1o en TIOBE). Ver *Cpp CSharp*
- *Hist. L.*: Creado en *Bell* por Dennis *Ritchie* (creador de *C*, *INode Hist. Linux*, *Turing A.*) & Kernigham'72 a partir del lenguaje B de *Thompson* para añadir F. a *Unix*.
 - ANSI estandariza 1º C, luego ISO. Hay Exten. para *Embeb.* y *RT*.
 - *Cpp* y *CSharp*: derivados
- Su *Portab.* permite: crear
 - Apl. *Embeb.*: *Microcontroladores*, *Arduino*. El otro es *Java*.
 - Apl. *Industr.*: como proc. texto.
 - *SO*: como *Unix* y *Linux*
 - *Cient.*: *Anal. Num.*: integrales sin primitiva...

30.2 Características generales

- Características:

- L. *Nivel* medio: Ventajas del alto nivel (cerca a humano) con bajo (*Acces.* Direct. o *Aleat.* a *Mem.* y *Dispos.*, *Punt.* y *Op.* a nivel de bit $>>$, puede *Llam.* a *Bibl.* escritas en *Ensambol.*).
 - L. *Compil.* entonces Cod. *Efic.* (*Vel.* y *Pot.* y *Compacto*). Inconv. *Depur.* compleja
 - L. *Propos.* General. Es L. *Tip.* no fuerte (permite excepciones y *Casts*)
 - L. *Parad. Progr.*: *Estr.* y *Modul.* (*Imper. Proced.*)
- Hay muchas *Bibl.* y *Compil.* (*GCC*, *MinGW*)
 - *Tip.* débil

30.3 Elementos tokens

- **Token** (no confundir con T. *Ring*): *Atom.* *Sintax.* o min. *Ud. Pars.esable*. Los 6 básicos son: *Id.*: *x*, *up Pal. Reserv.*: *if, return Separ.*: () *Op.*: + *Literal*: true *Coment.*: /*Func*/ Fig. 25.8. Ver *Vincul.*, *Instanc.*, *IP*, *PID*, *INode*, *Obj.*, *RFID*, *Cl.*, *Puerto*, *BlockChain*
- **Id.** (Identificador, no confundir con *Pal. Reserv.* o *Etiq.*): *Token* (Primit. *Semant.*) que nombra (*Vincul.* o *Simbol.*) entidades *Def.* por Programador como *Var./Cte*, *F.*, *Subrut.* *Modul.* (y *Tip. Dat.* como *Typedef (Struct)??*)
 - En C siguen *ANSI*: es CaseSensitive, <32 caract., permite Underscore *cierra_puerta()*. Ver *RFID*, *Suplant*.
 - \forall *Id.* \in *Domin.* (o Esp. Nombres) y \in *Ambito*.
 - Otro: ver *Instanc.* *PID*, *UID*,....
- Suelen ser *Unic..* Es *Unif..* Ver *Direc.* *Topos*, *SO*, *PCB*, *INode*, *Arch.* *Dispos.*, *SSID* (*Wifi*)
- **Pal. Reserv.** o Clave o Keyword: *Token* para el *Compil.* que no puede usarse como *Id..* En C en minúscula, ej: *break, if, case, struct,..* muchas de *Estr. Control, Tip. e Instr.* Sin embargo *printf* no lo es (ver T32Punt.).
- **Separ.** : *Token* delimitador. Permite *FBF*. Ej: Puntuación, tabulación (C dif. a *Python*). Separar Sentencias con ; (como *SQL*) y agrupadas por {}. Ver *Cod. Pref.*, *LISP*, *Comando cut. EOF* (End Of File) y \n (*Fich.*, *ASCII*, *gets (Punt., C)*)
- **Op.** : *Token* o *F.* que actua sobre 1 (*unAria*), 2 (*binario*).. n variables. Es un *Token*.
 - Aritm: +, -, *, %
 - Asign.: ++, --, =, + =, - =, * =, / =, % = todos los Artim con igual
 - Lógicos: >, >=, !=, &&, ||, !
 - Bits (Ventaja de C): &, |, (^XOR), (*ComplA1*), >> (*DesplBitDcha*)
 - Op. Ternario (?): equivale a *if-else. ExprBool ? ValorSiTrue : ValorSiFalse*.

Permite crear *Expr..* OJO prioridad: al combinar los anteriores en una sentencia normalmente de izda a dcha. Ver *EA*, *Estr. Dat.*, *Sobrecarg. Op.* (*Polimorf.*).

- **Literal** : Token o notación que representa un valor fijo. Concreción de un Tip. Dat.. Ej: `int a=1; string s="cat"` siendo 1 y "cat" los L. Cte: valor inalterable durante todo el programa. Cte simbólicas con Directiva (#define PI 3.14)
- **Coment.** : Token o MetaInform. para facilitar Compr.. Ej C: * Bloque*, Linea. Pueden Docum.

30.4 Elementos: descriptores de datos: tipos, variables y struct

- *Estr. Dat.* (Int. Dinam. No-Lin.) y *Tip.*:
 - T. Básicos: `char, int, float, double` y `void`.
 - T. Deriv.: `string=char[90], complex, Struct, Array`, datos-enumer.??,,
 - Modific. del Rango de Tip. Básicos (Precisión): `long, short, signed, unsigned` (doble solo long)

Ver IEEE 754

- **Var.** (Variable): nombre del Esp. que Almac. un Dat. Tip.. Se Decl. como *Tip.+Id.* ej: `int Juan`. a) *Dat.* (contenido cambiante pero de un Tip.) b) e *Id.* (o nombre) . Contrastá con **Cte** (no cambia). Hay:
 - Local: Ambito
 - Global: desaconsejadas (Acopl.). Ver Manej. Signal en `cchildUDP.c` en RT

4 formas de almacenarlas:

`static` (permanece en mem. durante todo T. Ejec., en Java es como global, si cambia uno cambia en todos)
`register` (la recomienda para poner en registro de CPU),
`auto, extern.` Ver Dinam.. Ver Punt., Vista. Etiq.

- *Struct: Estr. Dat.* Heterog.

30.5 Elementos: descripción de acciones: expresiones y estructuras de control

- *Expr.*: `double result = pow(x, y);`
- *Estr. Control: Condic. (if-else,switch). Iter. (for,while-do).* Ruptura: `break, continue, GoTo`

30.6 Estructura de un programa

- *Estr. Progr. Fuente en C*: Fig. 30.10. A groso modo: Cabecera y Main
 - Coment., Docum.: Vers.

- *Directiva y Macro*: Preproc. *Bibl.*
- *Decl.* F. y Var. Global: prototipos de F. Ej. *gstop*
- *Main*
- *Def.* F.: al final o en otro *Modul.*

30.6.1 Preprocesado, directivas y macros

- *Preproc.* : primer programa invocado por el *Compil.* que *Subst. Macros*, incluye otros archivos y *Directiva* (ej. en *C* reemplaza `#include <global.h>` por contenido de *global.c*, idem con `#define`), elimina comentarios en *Cod. Fuente*. Ver *BD*.
- *Directiva* de *Preproc.* o Pragma: *MetaInform.* (normalmente en *Cabecera*) que le dice al *Preproc.* del *Compil.* porqué *Subst..* Ver *D. Europ.*
 - D. *Bibl.*: `#include ".....common.h"`.
 - D. *Block*: `#define, if-else.`
 - D. *Macro Param.*: tipo de *Directiva* que empiezan por `#define` y le da el Poder *Expr.* de F. Ej: `#define PI 3.14159 #define pred(x) ((x*PI)-1)`
 - * *Macro*: *Instr.* formada por sencillas que se Ejec. *Secuenc.* (no *Estr. Control* como *for*). Patrón que especifica cómo debe asignarse una I. a una O. de *Subst..* Ej: *Directiva M.* con Poder *Expr.* de *F.*

Ver *Meta*

30.7 Funciones de librería y usuario

- Estandar *ANSI* es incompleto.
 - *Bibl.* estandar: con `#define`. Ej: *alloc.h* (mem. *Dinam.*), *Stdio.h* (*I/O*), *Stlib.h* (decl. varias), *string.h* (*Caden.*), *Stdio.h* (con *fseek*, para fich. ojo *fstream.h* es para E/S *Cpp*), *conio.h* (pantall), *io.h* (*I/O Unix*), *math.h* (*Matem.*). *uniStd.h* y *fcntl.h*. Las de *RT*
 - *Bibl.* de usr: compilables por separado

Se crea *Cod. Obj.* y luego se *Enl.*

30.8 Entorno de compilación

- *Cod. Obj..*
- Las 3 etapas: *Compil.*, *Preproc.* (*Directiva global.h*) y *Enl.*
- Make *Efic.* (-Wall)

30.9 Herramientas para la elaboración y depuración

- Editores: de *Fuente* con *Autocompl.* (IA), Autotab., Color de *Pal.* *Reserv.* y Visualizar func.
- **GCC** : *Compil.*: en modo texto de *GNU Linux* y *MinGW Win*, ver *Efic.*, *Enl.* En modo *GUI IDE*. **Make** : gestor de dependenc. para compilar autom. *Fuentes* ordenes para ello (Fig. 30.10)
- *Depur.*: GDB e *IDE*
- Otras: Gest. *Proyec.* Progr. (varios ficheros a la vez), Contr. *Vers.* y *Make*
- **IDE** (Integrated Development Environment, Ambiente de Desarr. Integr., no confundir con PATA): *Herram.* que auna todo lo anterior para *Simul.*. También:
 - Crear *Proyec.* Progr.. Editor (colores), *Autocompl.*, breakpoints
 - Est. de F. más usadas para *Efic.*. Ej: *MATLAB*
 - *Docum. Autom.* desde *Coment.* (Aguadulce NetBean Java)
 - *App Inventor* para *Scratch* y *Movil*
IDE Arduino.
 - Motor *Videojuego*,
 - NeatBeam (Aguadulce) *C Cpp* y *Java* , Spyder/Jupyter/Eclipse (*Python*), *Microsoft VisualEstudio* (no confundir con el de abajo) *VisualEstudioCode* (para Unreal Engine *Videojuego*, *Git*) y *BorlandC++Builder*. Los *SGBD* lo son??. Ver *CASE*

Token name	Sample token values
identifier	x, color, UP
keyword	if, while, return
separator	}, (, ;
operator	+, <, =
literal	true, 6.02e23, "music"
comment	/* Retrieves user data */, // must be negative

30.10 T31CGeneral

```

do {
    if(ReceiveTCP(cad)!=-1) //in blocking mode
        fprintf(stderr,"..... YOU SAID: %s ....\n\n", cad);
    fflush(stderr);
}while(strcmp(cad,"PARA")!=0);

// Name of Program → Documentation section

#include<stdio.h> } → Preprocessor Directives
#include<conio.h>

#define max 100 → Definition section
void add(); } → Global declaration section
int x=100;

int main() → main () Function section / Entry Point
{ int a=100; → Variable declaration

printf("Hello Main");
return 0; } → Body of Main function

void add(){ → Function Definition
    printf("Hello add");
}

typedef struct {
    unsigned char flags;
    unsigned char payload_type;
    unsigned int sequence_number;
    unsigned long timestamp;
    unsigned long synchronization_source;
    unsigned char payload[PAYLOAD_SIZE];
} rtpPacket;

switch(VADNum)
{
    case -1: //Finish the word
        SenderUDPQuickly(COMPRESSEDTOTALSILENT, N_TOTALSILENT);
        TotalSent+=N_TOTALSILENT;
        break;

    case 0: //There isn't voice
        break;

    default: //There is voice
        //Code get compressedframes to send
        code_two_frames(framebufferToSend, compressedframes);
        //Send compressedframes
        SenderUDP(compressedframes);
        TotalSent++;
        break;
}

CC = gcc
LFLAG = -lm -pthread -L ./recordlib -lportaudio
CFLAGS = -Wall -O3
PROGRAM = cfatherTCP

all: $(PROGRAM) clean

# Implicit Rules
.c.o:
$(CC) $(CFLAGS) -c $<

#Explicit Rules
$(PROGRAM): cfatherTCP.o cchildUDP.o ./clientTCP/clientembedTCP.o
coder/golay.o clientUDP/clientembedUDP.o VAD/VAD.o ../common.o
$(CC) $(CFLAGS) -o $(PROGRAM) cfatherTCP.o cchildUDP.o client
VAD.o common.o $(LFLAG)

clean:
    rm *.o

# Individual File Dependencies
feembed.o: FEFUNC.h fileio.h
FEFUNC.h: FEFUNC.h
fileio.o: fileio.h
pablio.o: pablio.h portaudio.h ringbuffer.h
coderembed.o: coderembed.h common.h golay.h q8.tab
golay.o: golay.h
clientembedUDP.o: rtp.h

```

Chapter 31

*32. Lenguaje C: Manipulación de estructuras de datos dinámicas y estáticas. Entrada y salida de datos. Gestión de punteros. Punteros a funciones.

31.1 Introducción. Conclusión

- Ej-motivador: ¿como hacer una *Lista Dinam.*? Sol. con *Punt.*
- Historia: mujer Yushchenko⁵⁵ *Punt.*, y C Ritchie *Bell*
- Futuro: no *Punt.* como *Java* (aunque referencias)

31.2 Lenguaje C

- Hist. y caract. de *C*: *Portab.*

31.3 Entradas salidas

- **Std io.h:** Bibl. Estand. de I/O porque C no tiene Pal. Reserv. de I/O como Python (*print*) Java¹. El Main inicia los siguientes Flujos: Stdin/out/err (ver Descr. Arch.). Ej: *fflush(Stdout)*; después de un *printf*. Veamos Instr.. Ver RT *uniStd.h*.
- *scanf(cadControl,listaArg-Param.):* permite R. datos *Form.* (indicamos *Sintax.*) Java²
 - %c (caracter-char), d (entero-int), i (enterooHexa), e (flotanteExpon), f (flotante), g (tomaEoFEIMasCorto), o (octalSiCeroInicial), s (cadena-Caden.), u (enteroSin-Signo), x (enteroHexaSinPref0x). Ojo a *Overflow*.

¹*System.out.println* es de Bibl. *java.io.PrintStream*

²*import java.util.Scanner; Scanner tecl= new Scanner (System.in); s= tecl.nextLine();*, para form. con *Expr.* regular??

- `int getchar()`: output char leido por teclado.
- `char *gets(char *cadena)`: lee cadena, le incluye char **Fin Caden.** 'barraInvertida+0' la situa en Direc. aPunt.ada por cadena.
- `conio.h`: ³
- `int printf(char *cadenaControl, listaArg)`: Form. similar a `scanf`, si `Err.` da n°<0, si no n° char impresos.
 - `puts` (similar) y `putchar` (char en pos. cursor actual): Parece que nuevos Vers. de C no recomiendan a estos ni a los `get`.

31.4 E/S mediante ficheros

- `FILE *fopen (char *nombre, char modo)`: abre-crea *Fich.* con el nombre y modo indicados y ver si OK (como *Fork*). FILE es un *Punt. Fich.* def. en *Stdio.h* Java ⁴ o *Descr. Arch.??*.
 - `int fclose(FILE *fich)`: cierra
 - *F.* de *Fich. Texto* (Tip. *Fich.*, ver): similares a las *Estand.* de antes: `fscanf` (R. Form.), `fgets()` (R. *Caden.*) `fgetc` (R. char). `fprintf,fputc,fputs()` (idem).
 - *F.* de *Fich. Binar.* (Tip. *Fich.*, ver): `fread` (R. conj. dat. de long. fija, bytes o chars), `fwrite()` (idem).
 - Otras: `fflush` (fuerza salida datos acumulados en *Buffer*). `fseek/ftell/rewind/feof` (pos. cursor situa/dice/comienzo/checkeaFinal).
 - *Separ. EOF*: *Err.* devuelto por `puts`. También `while (!feof(fp))...`

31.5 Manipulación de estructuras estáticas

- *Estr. Dat. Estat.* (contrario *Dinam.*): Def. antes de Compil. y no cambia en Ejec. Ej: *Array* y *Caden.* de Tam. Fijo. Ver *IP, Rut.*
- *Array* : de *Estr. Dat.* Homogenea o *Conj.* dat. del mismo *Tip.* *MultiDim..* El Compil. Asign. direc. mem. contiguas y el 1er elem. tiene la menor. Ej: *Multimedia*
 - *Vect.* (1D): `float v[3]=2.1, 3, -2.6;`
 - *Matr.* (2D): *Tabla*, Se puede *Ind.exar (Enumer.)* `A(2,4,19)`. Ver *Rot., Filos. MATLAB, Impr., Graf., Sparse, AES*
 - A. vs Tensor: en *Python NumPy's ndarrays=Tensor*. Except that tensors can run on *GPU* to accelerate computing.” [Pytorch Documentation]
 - *Topol.*: A. Esfera, Toro (se pueden sumar..)

Ver *Mem., AMX (Exten.), MicroProc. Vect. (Supercomput.), Caden.. Dinam.*

³`int getch()`: lee caracter a caract. pero no muestra por pantalla. `int getche()`: idem pero si muestra. Creo DesActual.

⁴`FileWriter f = new FileWriter("f.txt"); f.write("Hola" + 2); f.close();`

- **Caden.** o String Texto: la longitud *Decl.* arada debe ser $l + 1$ porque se debe incluir el char **Fin Cadena** 'barraInvertida+0'. La *Bibl.* *string.h* tiene:
 - *strcpy(arg1,arg2)* (copia), *strlen(arg1)* (longitud), *strcat* (concatena), *strcmp(arg1,arg2)* (compara),
 - *Stdio.h: gets/puts(cadena)* (R/W)

Ver *Array, I/O C, Estr. Dat. Homogenea. Caden. Block. Stream, Pila Protoc..*

- **Struct** (no confundir con *Dic.*, *Estr.*, ni con *Union* o *Enumer.* (abajo)): *Estr. Dat. Interna Comp. Estat. Heterog. Comp. de Var. de dist. Tip.* (llamados *Campos*). *Cl.* muy básica de solo *Atrib. (Her. reduc.)*.
 - *Recurs.*: El *Tip.* puede ser otra *Struct* y suele ser común definir tipo nuevo (para evitar struct cada vez):
 - *Pas.* por *Refer.*: *Acces.* mediante un *Punt..*. Dentro de F. se usa como *Str→ Campo* y no como habitualmente *Str.campo*.
 - *Array* de S. (Fig. 30.10):
 - *typedef*: permite *Def. Tip.* (o nombres alternativos). Ej: en *common.h* puedes *Decl.* en .c (*rtpPacket sendPacket;*).

Ej: Básico:

```
struct nombreStr{
    tipo nombreCampo1;
    tipo nombreCampo2;
    .....
}; //termina como Cl.
```

Ej: *Lista Enl.*

```
typedef struct NodoLista{
    char elem[50];
    struct NodoLista *Siguiente;
} Nodo;
//Funciones
int InsertarElemento (Nodo **); //evito struct NodoLista
```

Ej: Otras *Estr. Dat.*: *Arbol Binar.* (ver abajo)

Ver *JSON, XML*

- Otras:

- *Union*: similar a *Struct*⁵

```
union foo {
    int a; // can't use both a and b at once
    char b;
} foo;
```
- *Enumer.*: como una *Lista*, le pone *Ind.*

```
enum year {
    Jan, Feb, Mar, Apr, May, Jun,
    Jul, Aug, Sep, Oct, Nov, Dec
};
void main()
{
    for (int i = Jan; i <= Dec; i++)
        printf("%d ", i); //sale 0,1,2...
}
```

⁵en que es un conj. de *Tip.* distintos, solo puedes usar en cada momento uno de los elementos ya que el *Compil.* los pone todos en la misma posic. de memoria (el del elemento mayor).

31.6 Punteros

- **Punt.** : *Var.* que almacena *Direc.* de *Mem.* donde hay otra. Este referencia o apunta una celda o ubicación en lugar de usar el *NombreVar*. *Java* se elimina su complejidad y son Reference⁶

Inventado por ucraniana soviética Kateryna Yushchenko 1955 en L. **Address Programming Language** de los 1ºs L. *Nivel Alto* y 1º en permitir Mod. *Direc.* indirecto, pero desconocido en occidente y atribuido a Lawson 1964 *Hist. L. Op.*:

- &: da Direc. Mem. de Var. en Hexadec.
- *: da valor al que apunta el Punt.

Ej: Fig ??

```
int Var=10; //crea Var con valor 10
int *ptr=&Var; // crea puntero que apunta a direc. de Var
int a=*ptr; //asigna a a el valor apuntado por ptr
```

- **Null** : valor inicial de *Punt..* Es bueno que apunte a este o a algo para evitar P. Loco. *Hoare* (*Ord.* Quicksort) se disculpó por inventarlo *Hist. L.*, pues ha traído muchas *Vulner.* y Rupturas de *Sist.* como *Fall. Segment.* (=Violación de *Acces.* que da *Signal POSIX SIGSEGV*) por **Null Pointer Dereference**.
 - * *Buffer Overflow: Vulner.* relacionada con *Fall. Segment.*=Violación de *Acces.* (y con *Null* de *Hoare* (*Ord.* QuickSort)) e *Intr.*. Usada por Gusano Morris (*Malw.*)

Ver *NaN, SQL, Lista Enl. Mod. Rel., Dat. Perd.*).

- Ej: *Array* (ver abajo), *F.* (abajo), *Struct* y además *ClAj* o *Ln*

Es *Unif.* por dar *Dinam.* (*Lista Enl.*). Ver *SP CODE2* (*P. Pila*), *Blockchain*, *Ind. Sparse* (*BD*).

- *Op.* con *Punt.*: las únicas permitidas son **sumas,incrementos y comparac** +, ++, -, --, <, >
 - Permite Mod. *Direc. Offset* , o Indirección (*INode*) y construir *Estr. Dat. Dinam.* (*Lista Enl.*).
- *Punt.* a *Array*: en *C* los A. se *Pas.* por Ref. *v[0]* es la 1^a *Direc..* Veamos *Equiv.* de Op. *Acces.* por Punt. vs *Ind.exado*:

```
int v[5]={1,3,5,7,9}; //vector de 5 elementos
int *ip; //ptr a int
///////////
ip=&v[0]; //equiv. a ip=v;
x=*ip; //x=v[0];
*(v+1); //v[1];
v+i; //&v[i];
```

- **Punt. a F:** similar a *Array nombreF=Direc. comienzo de Subrut.=Punt.* como muestra *Sintax. de Cabecera: tipo (*nombreF)(tipo Param.1, tipo Param.2,..)*; Ej: parecido a *Progr. Fun.*

⁶Reference: si imprimimos un objeto se muestra su dirección ecript-Hash dentro del *Esp. Mem.* (*Heap*) de *Java*. *Demo D1 = new Demo(); System.out.println(D1); //sale Demo@23452*

```
int doble(int x){ return x*2; }

void main(){
    int (*duplica)(int)=&doble; // le da direc. de F.
    int x=(*duplica)(3);
    printf("%d\n",x);
}
```

31.7 Asignación dinámica de memoria

- *Asign. Dinam.* (contrario a *Estat.*): reserva de Mem. *Decl.* sin tamaño Estat. por flexibilidad o versatilidad de los *Graf.* donde *Ord.* y *Rel.* de elem. cambia. Los F. principales son:
 - *malloc*: asigna los Bytes especificados
 - *realloc*: aumenta o dismin. *Bloq.* mem.
 - *calloc*: como *malloc* pero inicia a 0 y especificas nº Bytes
 - *free*: mem. *Libre Basura*
 - *Punt.*: permite hacer A. D. en L. como *C* que no lo permiten de forma nativa
 - Permite ej: *Insert./Delet.* de *Estr. Dat.* D. por *Lista Enl.* como *Fich.*, *Arbol*, *Tupla* o *Array Din.*

Ej: *Vect.* con *sizeof* Fig. 31.10 crece al *Insert./Delet.* datos

```
int *v;
v=malloc(30*sizeof(int));
```

Ej: *Matr.* con *Punt.* Fig. 31.10 Es *Unif.* por *Lista Enl.*. Ver *Segment.*, *Rut.*, *S/DRAM*, *Prior.* D. (*Plan.* *RT*)

31.8 Arrays, Listas,..

- *Estr. Dat.* básicas:
 - Celda *Lista*: con *Struct* simple (siguiente), *dobleEnl.* (sig. y ant.), *dobleEnlPrior*. (idem+Prior.). Fig. 31.10
 - *Vect. Dinam.*: con *malloc* (ver arriba).
- *Lista Enl.*:
 - *Pila*:
 - *Cola*:
 - L. Doble Enl. y *Prior.*: Fig. 31.10
 - L. de L. Fig. 31.10.
- *Arbol*: cada Nod. a *Punt.* a varios
 - A. *Binar. Busq.*:
 - A. *Equil.*

Arbol Binar.

```
typedef struct NodoArbol{
    char elem[50];
    struct NodoArbol *hijoIzdo, *hijoDcho;
} Nodo;
```

- *Graf.os: Matr. Sparse*
- *Hash:*

31.9 Gráficos y videojuegos en C (solo SAI)

- *Videojuegos, Render., 3D*
- *graphic.h: Bibl. 3D BGI (Borland Graphic Interf. de MS-DOS Win. Como P5.JS JavaScript, puede trabajar en modo CLI o GUI. Crea Socket (ver abajo). Para Videojuego, Imag. Vect.. Otras son OpenGL (Render. Libre). Java*⁷

```
#include <graphic.h> //
initgraph(driverTarjGrafElegida, modoResol., dirPathBiblBGI);
printText;
printLine;
printCircle;
closeGraph;
```

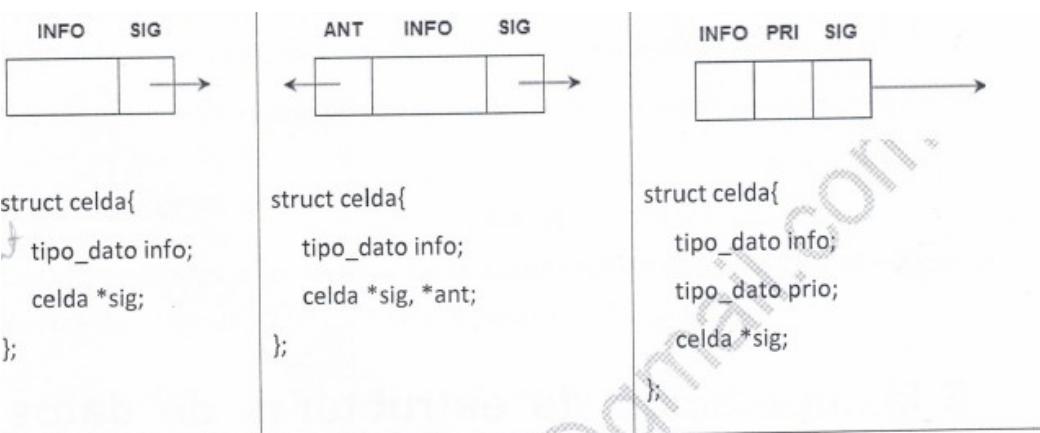
⁷Processing, ver *JavaScript*

ptr 10 edad

0x503fb43

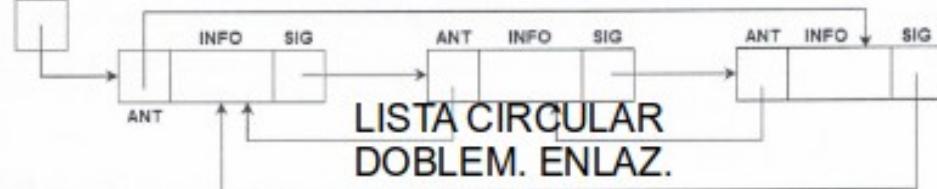
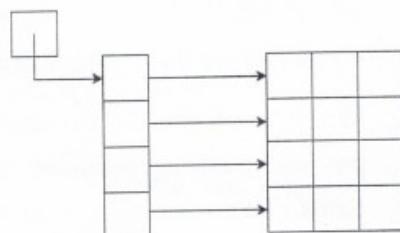
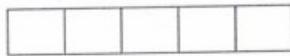
10 edad

31.10 T32CEstrDinEstIO

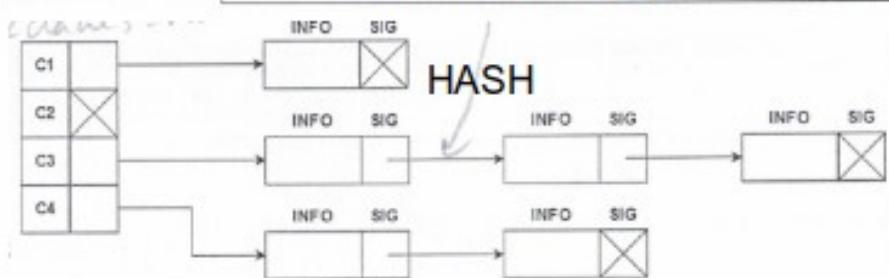
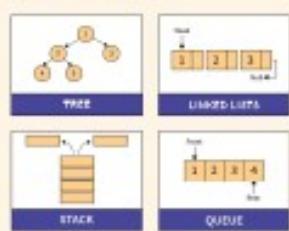


```
tipo_dato *m;
m = malloc(tam * sizeof(tipo_dato));
```

```
tipo_dato **m;
m = malloc(nfil * malloc(30 * sizeof(tipo_dato*)));
for(int i = 0; i < nfil; i++)
    m[i] = malloc(ncol * sizeof(tipo_dato));
```



E.D. BASE



Chapter 32

Común BD

32.1 Persistencia de objetos (T29Borrador)

- **Persist.** : hacer que los datos se puedan recuperar una vez modificados (deshacer o dar *Backtracking*). En caso *Fall.* del Sist. o *Transac..* Como la *Cache* o la *Memoization* (*Efic.*), acelera la *Carg.* y *Transm.* de cualquier *Obj..* Es *Ortog.* o *Transpar.* cuando se hace *Autom.* en cada *Actual.* de *Dat*, para ello se debe cambiar *Map.* (o *Estr.* o *Form.*) del *Form.* *Progr.* al *Form.* *MS*, i.e. se *Serielizar* (ej. *JSON*, creo sinonimos *Persist.=Serielizar*).
 - Arq.: Técnicas: a) , b) c) *Journal*, d) Mem. No *Volatile*
 - SO: *Hibern.* (*Imag.* *Disco* o *USBLive* que mantiene *Config.*), *Maq.* *Virt.* (*Est.*). *Interr.*
 - L.: *Progr.* *Compo.* (*Jakarta*)
 - BD: *SQL/PSM Proced.* *Almac.*, *BDOO* (*3D*). Ver Ej. con Hist. *Trigger*. Comando *MySQLi* (ver abajo)
 - Red: *AJAX* (*Serielizar JSON*), *ActiveX* (*Progr.* *Compo.*). Comando *PHP SQL MySQLi* crea un *Obj.* P. (ver ejerc.)

Ver *Almac.* *Masiv.*, *Recol.* *Basura*, *Gest.* *Mem.*, *Transac.*, *Backend* (C. *Acces.* *Dat.*). *Preserv.*, *IRP* (*Petic.*, *Win*). *SSD DRAM*, *Fich.* *Tmp*, *Kubernetes*

- **Transac.** : *Ejec.* de un R/W (*Acces./Actual.* o *Trigger*, *Op.* *BD*) a una *BD* o *Fich.* de forma TOTALMENTE correcta (sin *Interf.* y *Coher.*) en un entorno *Concurr.* *MultiUsr* (cuando hay varias T. a la vez). Ej: *SSHFTP* debe cuidarlas. Se debe cumplir:
 - *Atom.*: op. totalmente correcta o incorrecta (pero no a medias). Se usan técnicas de *Journal*, *AcRe*
 - Otras: *Integr.* (*Consist.*), Duradera (*Persist.*). Se usa *Excl.* *Mut.* del *Recurso* (en Sec. *Crit.*)
 - *TCL*: es un SubL. de *SGBD* y *SQL* relacionado con *DCL*.
 - *OLTP*: On-Line Transaction *Proc.* usado en *Sucursal*.
 - **CICS** (Customer Information Control System o Monitor de Teleproceso): *Middlew.* de *IBM*, permite *Concurr.* de proc. de *Usr Acces.* a *BD*, para *Transac..* Ej. escrito en *COBOL* para *Mainframe IBM* de *Sucursal Bancos*

Ver *Commit*, Sec. *Crit.*, *CICS*, *SEC (Sucursal)*

- *Campo* (field): de *Mem.*. Ver *Form.ulario (Apl. Web)*, *Comando cut*, *Atrib.*, *Lista Enl. Struct*, *Key. Atrib.*.

32.2 35. La definición de datos. Niveles de descripción. Lenguajes. Diccionario de datos.

- *Dic. BD* : conj. de *Tablas* con *MetaDat.* (*Rel. Tab.*, *Format.*, *Uso*,). Las *CASE* son una *Exten.* del Dic. BD. Contiene el:

- *Esquema BD*
- *Ind.*
- *Proced. Almac..*

Ver *Dic., Arq. ANSI/SPARC*

32.3 36. La manipulación de datos. Operaciones. Lenguajes. Optimización de consultas.

- *Opt. Consult.* : ya que *SQL* es *Decl.* (no es *Alg. Rel.* ver), hay que elegir el *Plan.* de *Consult. Opt.* entre los muchos posibles.

Permite *Busq. Vel.* o *Consult. Masiv.* a gran *Vel.*. Técnicas:

- *Ind.* adecuado
- *FN*: por crear *Ind.* más rápido
- Factor de Join Selection adecuado
- *BD Distrib.*: decir nodos donde encontrar (*Topos*, *ISAM*)
- \nexists *Redund.* Log. pero si Fis. (por *Disponib.* u *QoS*). Ej. *SAM* e *Ind.*

Ver OLAP (MultiDim.)

32.4 37. Modelo de datos jerárquico y en red. Estructuras. Operaciones.

- *BD Jerarq.* (\subset *BD Red* \subset *BD Naveg.* abajo): *Model.* BD donde los *Arboles* de *Registr.* donde cada R. tiene un solo padre [NetworkModel] (en *BD Red* varios, *Graf.*). *Graf.* donde *Nod.=Registr.* y *Arco=Rel.*

- Ej: [BDJerarq] sea el padre *TablaEmpl(idEmpl,nombre)* y la tabla hijo para uno de los registros Empl. *TablaComp(CodCompon,Nombre,idEmpl)* el campo *idEmpl* es siempre el mismo n° 100, cada Registr. empl. tendrá su tabla.
- Ej: *DNS* y el *IBM IMS* (Information Management System)

- Es el 1er modelo *Hist. BD* implementado: creado por *IBM* en 1960, olvidado por éxito del Mod. Rel. de *Codd* y resurgido con llegada de *XML Hist. BD*. Hoy usada para *Almac. Dat. Geográficos y Sist. Arch.*.
- BD *Naveg.*: engloba las BD Jerarq. en Red. Es de Bachman *Turing Hist. BD*
- Multi*Dim.* y OLAP ((On-Line Analytical Processing)): *Tabla Multidim.* o de *Array*. Toma lo mejor de las *BD Jerarq. Naveg.* y Rel. Ver *3D, Opt. Consult.*, Fig. 33.11.
- ***BDOO*** (BD Orientada a *Obj.*): *SGBD* donde la Info se representa como Obj. (en el sentido de *POO*). “Automágicamente” (*Transpar.*) se hacen consultas como: BD.field. Con *CSharp, Cpp, Java*.
 - Un Obj. tiene *Atrib.* y una Instanc. es un relleno de Dat. de dichos Atrib. (una Column.). Los Atrib. de un Obj. se pueden Rel. con los de otro (ObjReport.Code=ObjActiv.Code). Obj. similares pertenecen a la misma Cl. (Creación por *Her.*). Cl. similares forman una *Jerarq.* (ver *Servic. Dir.*)
 - BD **Obj. Rel.**: es un subtipo de BD Rel. con Dom. no Atom. (ver 1 *FN*). La mayoría de SGBD modernos lo son. Ej: *Oracle DB, SQL Server..*
 - *Map.* o *Asign.* Obj. Rel. (Object-Relational mapping ORM): Paso de una *BDOO* a una *Rel.* permitiendo la *Persist.* (es conversión de *Tip.* Ej: Jakarta JPA). Crea una *BDOO Virt.* sobre una BD Rel. Una *BDOO* extiende los L. Dat. persistentes de forma *Transpar.*.
 - *3D*: las *BDDO* se usan en *Biolog. Molecular* o *BD Esp. aciales*.

Ver Obj. *Distrib.* (*Progr. Compo.*)

32.5 40. Diseño de bases de datos relacionales.

- *Model.* ***ER*** (*Entidad Rel.*, no confundir con *Mod. Rel.* de *Codd70*): de [Chen76] (no es Turing A. *Hist. BD*). *Diagr.* UML de una *Ontol.* (visión general y clasificaciones de los términos utilizados y sus *Rel.* para un determinado ámbito de interés). Permite *Dis.* una BD y se obtiene del *Anal.* de *Req.* o *Semant.* (lectura). Los 3 pasos son:
 - *Entidad* y Rel.: ver *Atrib., Obj., Autent.*
 - *Graf.* Relacional: paso Mod. **ER2Rel.**: es un functor *Migr.??* a) Toda E se transforma en Tabla. b) Atrib. en col. c) El Atrib. Clave es la Clave Primaria. <https://desarrolloweb.com/articulos/paso-tablas-entidad-relacion.html>.
 - *FN*

32.6 42. Sistemas de base de datos distribuidos.

- *Arq.* Multi*Tier* (o Multi*Capa*, no confundir con *Arq. ANSI/SPARC* ver): general. del Client.-Servid. para sist. en *Red* o *Distrib.* como: *Web* o *Progr.* o *Servid. BD*. Da las ventajas de la *Modul.:*
 - *Modul.:* *Transpar.:* cambios en *Backend* no afectan al *Frontend*:

- *Segur.* pues cualquier *Petic.* a la BD se hace mediante un *Servid.* que conecta con un SGBD. Similar a *Proxy*.

Según nº de Tiers distinguimos:

- a) De 1: Motor (Engine, *SGBD*) y el main están integrados (ej. *SQLite*)
- b) De 2: *Client.* y Servidor (que ejecuta el SGBD). Ej: *MySQL*
- c) De 3: *Presentación/Usr* (navegador *Web*), *Serv. Negocio/Apl.* (*Apache Tomcat* con *Applet Servlet*) y *Datos* (BD como *SMTP*), *Jerarq. Red.* Ver Fig. 33.11

- *Frontend, Backend*

Ver *BD Nube, T. Proveedor*

- ***BD Distrib.*** (no confundir con *BD Nube* abajo): es uno de los 2 tipos de *Almac. Distrib..* Para *Consult. Masiv.* Distrib en *Cluster* garantizando Alta *Disponib..* Pueden ser *SQL* o *NoSQL* (ver ej.)

- IMPORTANTE *Almac. Distrib.* en *NoSQL* ha crecido pero según *Tma CAP* solo 2 de las 3 se pueden garantizar [*SGBD*]
- Ej: *Apache Cassandra* o *MongoDB*: *NoSQL*.
- Ej: *Apache Spark*: *SQL* (no confundir con *Arq. ANSI/SPARC*):
- *S. Paral./Distrib.: S. Escal. Vert./Horiz.*
- Ej: Distrib. el *Client.* lanza una *Petic.* (*Consult.*) y el *Servid.* de *Apl.* indica los *Nodo* donde están los *Dat.* repartidos de forma *Transpar.* (como *ISAM??, Topos=Opt. Consult.*). Ej: *DNS* o *Apache Spark* (*Consult. Masiv.*). Ej: *Arq. MultiTier* y *NoSQL* o *BD Nube*.

Ver *BD Nube*.

- ***BD Nube*** (no confundir con *BD Distrib.* arriba): *BD* que corre en *Comput.* en *Nube*. Como Microsoft *SQL Azure* o *Amazon DB Service*. Según mod. distinguimos *SQL* y *NoSQL* y según *Despl.* 2:

- *DBaaS* (DB as a *Servic.*): el *Usr* de *Apl.* (*Admin.*) no instala ni *Manten.* la BD, esto lo hace el *Proveedor* que le cobra al *Usr* según uso del *Servic.* También se cuida *Escal.* y Alta *Disponib.*
- *Maq. Virt.*: el *Usr* de *Apl.* se encarga de instalar y *Manten.* varias *Instanc.* (MV donde corre la BD en un *Servid. BD*) y el proveedor cobra t. de uso del *Servid.*

Ver *Almac. Distrib., MultiTier*.

- *BD NoSQL* (o *NoRel.*): *Dat.* sin *Tablas* ni *Estr.* fija. No permiten *Join*. Suelen ser *Dat. Masiv. Escal.* horizontalmente y que se *Manej.* con *IA*. Cumplen *Tma CAP* (ver *BD Distrib.*). Crecen con las 3 grandes *Empresa Tecn.* (*Google, Amazon* y *Facebook*).

- Tipos de *Almac. Masiv.*: a) *Dic.* b) *Docum.*: en *Estr. Dat.* como *JSON* o *XML*. c) *Graf.* d) *Orient.* a *Column.*
- Tipos: a) *BD SQL*, b) *Conj. Dat.* (no necesariamente computacional) como *Biblioteca*. c) *Corpus conj. texto o voces* (más de lingüística) Ej. *MNIST* (digit. mano a texto).

- Ej: *XML* [SGBD]
 - Ej: *Apache Cassandra* (*BD Distrib.* tipo de *Almac. Distrib.*: (ver *MultiTier*).
 - Ej: *MongoDB*
- *Servid. BD* : *Servid.* que proporciona *Servic.* de *BD*. Usa *Arq. MultiTier o Client. Servid..* Los *SGBD* suelen proporcionar los *Servicios* y permiten su *Admin.* Ej: *MySQL* y *Oracle*. Trabaja en *Puerto TCP 1433* (ver *Malw. Gusano*). Ver *BD Nube Proxy*

32.7 44. Técnicas y procedimientos para la seguridad de los datos.

- *SQL InYec.* : *Ataq.* que aprovecha *Vulner.* Añade Cod. SQL en la URL paso GET para una *Web PHP* con *Form. HTML* para *Autent.* (ver).
 - *Blog WordPress*: UrlWordPress/wp-admin te abre *Autent.* al *PhpMyAdmin* y luego I. SQL

Es *Unif.* BD+Red. Ver *Morf..*

Chapter 33

+34. Sistemas gestores de base de datos. Funciones. Componentes. Arquitecturas de referencia y operacionales. Tipos de sistemas. (6OctJoaq)

33.1 Introducción. Conclusión

- ¿Como funciona y que nos permite un SGBD?
- Hist. MySQL comprada recientemente por Oracle y sale MariaDB, 4GL (L.), Arq. ANSI/SPARC ANSI75
- Futuro: a) IA Anal. Masiv. b) Nube Stable Difusion (VA Amazon AWS). Dockers y MariaDB Kubernetes

33.2 Base de Datos

- **BD** (Base de Datos): colección *Ord.* de *Dat.* *Almac.* y *Acces.* (*TAP*) *Electr.* Es una *Estr.* *Dat.* Ext. (en MS) y posee las prop. de los *Fich.* (**InDepend.**, **Log.** **Fis.** ver *Arq. ANSI/SPARC*)
 - *Mod. Rel.* es el más *Popular* y dando:
 - a) *Esquema* (*Dic.* *BD*, *Restric.*)
 - b) L. *Consult.* y Alteración (*Op.* *BD*)
i.e. a diferencia de un Excel tiene *Rel. Tab.*.
 - Ej: *Bibl.* de libros o *Music.*, *IA* con BD, clásica Proveedor-Pieza-Envio (*Rel. Tab.*), *Google Map* (*AJAX*). *ACL* (*Autent.*)

Ver *Sist. Arch.*, *LDAP*, *Arbol B+.*, *ICANN*

33.3 Sistema Gestor de Bases de Datos y funciones

- **SGBD** (Sist. Gestor *BD*) o DBMS (DB Management System): [Begg] conj. de *Softw.* que permite al *Admin. Desarr.* una *BD* (*Def.*, crear, *Manten.*, controlar *Acces.* la *BD* y *Manten.*).
También es el *Softw.* que interacciona con *Usr* en el extremo. Ver Motor=SGBD (abajo)
- Las F. de los *SGBD*: aparte de las 2 de *BD* (Creac. y Consult.) resto sacar de TAP:
 - *TeleAcces.*: en *Red QoS*
 - *Vel.* de *Acces.* (*Opt. Consult.*): \nexists *Redund.* Log. pero si Fis. (por *Disponib.* u *QoS*). Ej. *SAM* e *Ind.*
 - *Backup Segur.*: *Preserv.* y *Persist.* (recuperar) (*Integr.*, *Confid.* y *Disponib.*) (*Permisos*)
 - *Concurr.*: Multi*Usr/Apl.* atendidos con *Cohes.* (ej. *Transac.* con *Excl. Mut.*)
 - *Present. Compr.*ensiva: amigable y entendible al usuario (Informes)

33.4 Tipos de modelos de BD

- **Model.** *BD*: Lenguaje sobre *BD* que a) *Descr.* su *Esquema*. b) Permite las 4 *Op. BD*. c) *Restr.* para *Integr.*. Por antiguedad:
 - *BD Jerarq.*, *Red*, *Naveg.*: Bachman60
 - *Mod. Rel.*: Codd70
 - M. *ER*: Chen76 y *OLAP* (*MultiDim.*)
 - *BDOO* y *Obj. Rel.*: 80
 - *BD NoSQL*: 90
 -
 - *BD Distrib.* y *BD Nube*

Ver *Cohes.*, *Dis.* Top-Down, M. *Aprend.* *Autom.* o *Prob.* o *Est.*, *Fuente Inform.*, *Conoc.*, *Descr.*, M. *3D*, M. *Actor* (*Calc. Pi*). M. *OSI*

33.5 Arquitectura de referencia

- **Arq. ANSI/SPARC** (no confundir con Apache Spark *BD Distrib.* ni Arq. Multi*Tier* ni competidor de Arq. *x86*): propuesta de *ANSI75 Hist. BD* nunca terminada pero es la Arq. *SGBD* de referencia (*Estand. Facto*), basada en 3 *Capas* (Fig. 33.11). Hay 3 *Esquemas* y *Vistas* del *SGBD*:
 1. E. Externo: V. *Usr* o Varios *Usr* a misma *BD*, que nunca es total (*Informe*, *Consult.*,...)
 2. E. Conceptual: V. *Admin.* (DBA) que Def. el *Dic. BD*

3. E. Físico: V. del SO y Hardw. del modo de *Almac. Distrib.* (ver abajo SAN, NAS y DAS)

Transpar. o *InDepend.. Log.-Fisic.*: los *Registr.* no coinciden en *Ord.* con *Direc.* Mem. Ej: *Mod. Rel.* e *ISAM*. Ventajas de la *Modul.*

- *Reus.* de la BD: pues no hay que reescribir las Apl. que acceden a ella.
- El modo de *Almac.* no influye en su manipulac. física *then* usr no modif. sus Apl. por cambios en almac. físico. *Indep. Log.*: las *Op. BD* no repercuten en Usr/Apl. que están accediendo a subconj. de los mismos

33.6 Arquitecturas operacionales

- Distinguimos:
 - A. Log o Fisic. Centralizada: todo se ejecuta en 1 Sist. *Informat..* Ej: BD MonoUsr desde PC o MultiUsr desde distintas Terminales
 - A. Fis. *Almac. Distrib.*: DAS, NAS y SAN
 - A. Log. *MultiTier*: *Client.-Servid., BD Distrib., Servid. BD*

33.7 Componentes de un SGBD

- *Jerarq.* de SubL. (*Form.*) o F. de SGBD
 - *DCL*: **GRANT,REVOKE**
 - *DDL*: **CREATE, ALTER, DROP**
 - *DML*: **INSERT, DELETE, UPDATE**
 - * DQL (*SQL*): **SELECT**
 - *TCL (Transac.)*: **COMMIT (Commit)**

Los más practicados son DDL, DML y SQL

- *Compo.* de *SGBD*: los siguientes SubSist.: RESUMEN Fig. 33.11: Ver Usr, DBA, Apl. y BD, el *SO* ayuda al SGBD (dando *Servic.*) y las *Consult.* se *Compil.* y pasan al *Opt. Consult.*
 - S. Motor o Engine: convierte *Op. BD Log.* en Fisic. A veces sinónimo de *SGBD*. Las *Estr. Dat.* de Almac. más Popul. son *Arbol B+*, *ISAM* y *Hash*.
 - S. Def.: *Dic. BD DDL* solo Accede el DBA
 - S. Manipul.: *DML (INSERT/DELETE)*
 - S. Gener. Apl.: *Conect. BD* (ver abajo), ayuda a crear ej. *Web de Acces.* con *APIs*
 - S. Admin.: dar *Segur.*

33.8 Herramientas de desarrollo de aplicaciones

- *Herram.:*
 - *Conect. BD*: Generador de Apl.: facilita la creación de Apl.
 - Consola (*CLI*): de 4GL (*Hist. L.*) para *Acces.* a BD
 - *PhpMyAdmin*
 - Generador de *Informe* : presentar parte de los datos en formato para pantalla o *Impr.* Ej. *Oracle Reports*
 - Precompilador (*Preproc.??*): traduce SQL a C (usando *Bibl.*) dentro de C (L. Anfitrión) los comandos *SQL* (L. Huesped).
 - *CASE*: crear *Esquema* e *Impl.* BD (*Exten.* del *Dic. BD*).
 - *CICS*: para *Concurr.* en *Transac.*.
 - *Migr. Esquema* con *FQL*
 - Herram. *Admin.*: para almacenar-recuperar, *Segur.*, opt. Consult., control de *Concurr.* y cambios. Ej: *PhpMyAdmin* en *MySQL* o *Oracle Enterprise Manager* en O.

33.9 SGBD Populares

- SGBD *Popular*:
 - *MSAccess* : inspirado en *QBE*. Para *Win*, muy *GUI* y para principiantes. No funciona con *Masiv.*. **Base** es vs *Libreoffice*.
 - *Oracle DB*: [Ellison70s] encargo de la CIA y se inspiró en paper de *Codd* (*BD Rel.*) *Hist. BD*. Es *BDOO* Obj. Rel. y muy usado. Soporta BD *Masiv.*
 - *MySQL*: ver abajo
 - *SQLite* : SGBD integrado o de 1 *Tier* (*Servid. BD*)
 - *PostgreSQL*: (nace del *QUEL*)
 - *DB2*: de *IBM* y *SO z/OS*. Soporta *Consult.* sobre *XML* y *JSON* (*NoSQL*). Ver *VSAM*.

33.10 MySQL, MariaDB y PhpMyAdmin

- *MySQL* : My (hija de [Widenius] *Hist. BD*) + *SQL* de *Oracle* pero Open Source (*Libre*, aunque ya tiene ciertos códigos cerrados)
MariaDB: versión Libre Libre de *MySQL*
Es un *SGBD* y *PhpMyAdmin* es un *Servid. BD* (de 2 *Tier*) de *Admin.* a través de páginas *Web*
Comando *mysqli* (*Persist.*)
Viene en varias *Stack Soluc.* (XAMPP). Workbench saca el *Model. ER*.
- *PhpMyAdmin* : se instala con XAMPP *Stack Soluc.* y permite *Admin.* una BD *MySQL* via *Web PHP*. Ver *Blog Ataq.*, *MySQL, Admin.*

¿Como manejar MySQL desde el *CLI*? Comando *mysql*

```
CREATE DATABASE mydb;
USE mydb;
CREATE TABLE books (...);
DESCRIBE books;
```

33 Database Management System

34. Sistemas gestores de base de datos Funciones. Componentes. Arquitecturas de referencia y operacionales. Tipos de sistemas. (6OctJoaq)

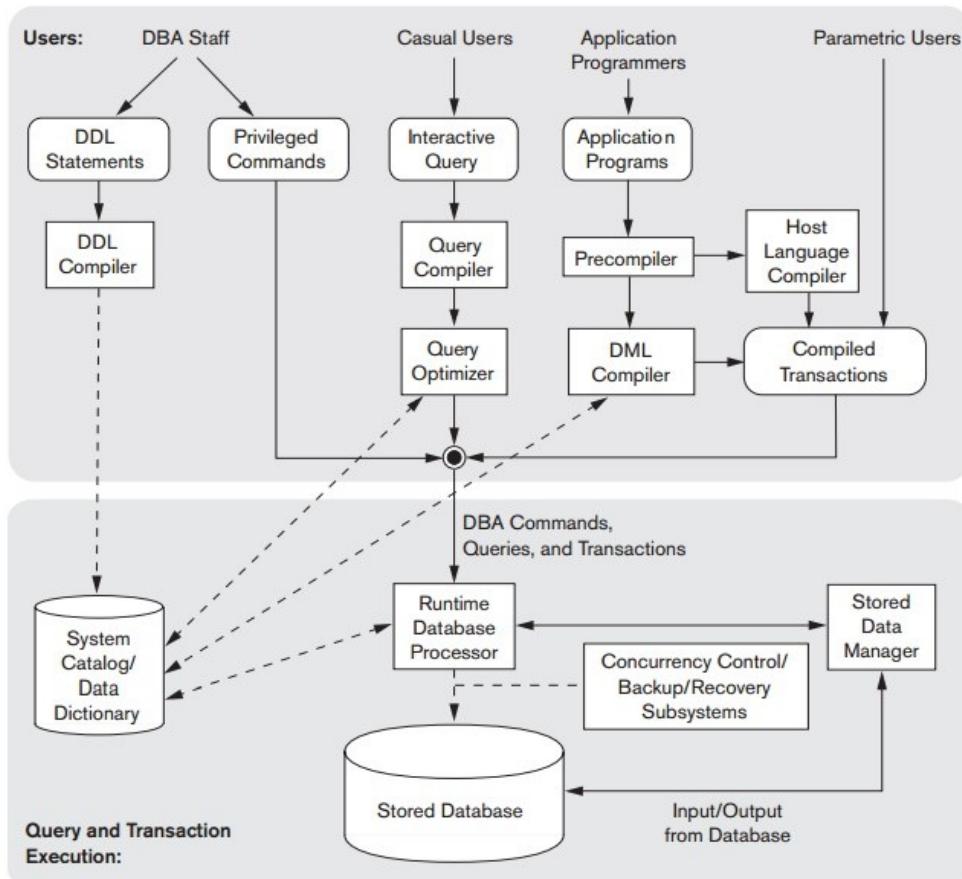
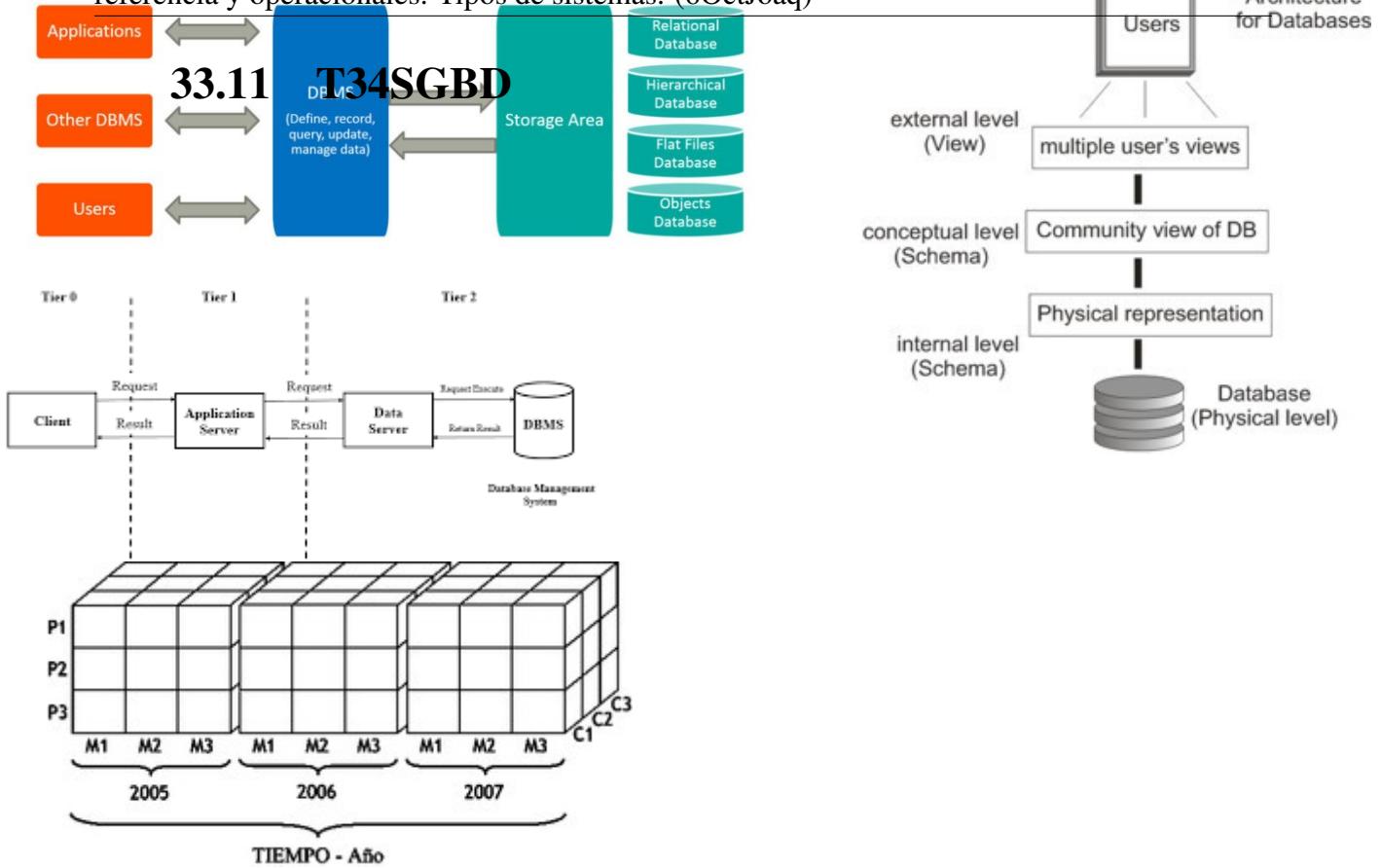


Figure 2.3

Component modules of a DBMS and their interactions.

Chapter 34

+38. Modelo de datos relacional. Estructuras. Operaciones. Algebra relacional. (10NovFuen)

34.1 Introducción. Conclusión

- Ej: ¿Poder Expr. máx de *SQL*? Ej: Relaciones matemáticas. BD Fincas
- Hist. [Date04] *Rel. Tab.* (Proveedor-Pieza-Envio), Boyce-Codd (4FN) Codd (Tma y Alg. Rel.) Lacroix
- Futuro: *FQL*

34.2 Base de datos y fundamentos

- *BD*
- *Matem.*: T. *Conj.*: *Conj. Tupla, Rel.*

34.3 Modelo Relacional

- *Mod. Rel.* : [Codd70] mod. teórico de *BD* (y *SGBD*) más *Popular* basado en *Log.1* (o de *Predic.*) y T. *Conj.*.

Todos los Dat. son *Tuplas* (=Registr. pedagógicamente hablando) que agrupados dan *Rel.* (=Tabla) entendidas en sentido *Matem.* (Rel. N-Aria) habiendo Rel. de Rel. (*Rel. Tab.*). Consta de:

- *Esquema=DDL*
- *L. Consult.=DML* (los 2 de *Codd*).

Además:

- Cumple: a) InDepend. Log. Fis (*Transpar.*), b) Decl. (“el qué”)??, c) Otros de *BD* (Flex., Unif., Sencill.).
- Permite *Impl.* una *BD* desde el *Model. ER*.

34.4 Estructura: el esquema

- **Esquema** BD: parte del *Mod. Rel.* para def. con el *DDL* la *Estr.* y *Metadatos*. guardado en el *Dic. BD*.
 - *Tabla, Tip. Atrib., Rel. Tab. y Restr. (Semant. y Sintax.)*
 - *Vista, Trigger*
 - *Arq. ANSI/SPARC* 3 tipos de E.

Ver CASE Umbrella (*Cooper.*), *Patr.*

34.5 Elementos básicos de una relación o tabla

- Definiciones: *Estr.* de *BD*=*Estr.* *Log.* *Fich.* (ver)
 - *Rel.=Tabla* : o RelVar. T. es una Rel. donde se permite *Tuplas* duplicadas (salvo *FN*). En la Fig. 34.14 podemos ver la Rel. *Trabaja-para* como *Correspond.* entre *Conj.*. Hay *MultiDim*. Ver Grupo (*EA*), T. Decis. (*Descr.*), T. Mem. (*Pagin.*), *PCB*, T. V. *Log.* *Rut.*, 4 T. Básicas *SO*. Ver *Array*. *FAT*
 - *Atrib.* =Col.: o *Campo* o *Conj.* o *Tip.* o *Domin.* Simple: conj. valores *Atom.* (ver 1 *FN*, D. Compuesto) permitidos, Ej: D. “Estado Civil”= {*Casado, Soltero, Separado*}, D. “Nacionalidad”={*Espaa, Francia, ...*}. NOTA: Un A. es una propiedad que representa una *Entidad* mientras que un D. es el conj. de valores que pueden ser asignados a un A. Ver *Rel. Tab., Cl., Fich. (Permitido..)*. *MetaDatos*
 - *Tupla=Fil.*: o *Registr.* o *Record* o *F.* porque *F(fila) = valores* o *Instanc.* de Atrib. Es *Dinam.* (*Insert./Delete*)
 - Otros:
 - a) *Valor=Elemento*: o Celda o Casilla o Caja. Puede ser *Nan* o *Null* por Dat. *Perd.* o por sinsentido *Sintax..*
 - b) *Card.*: C. Rel. (nº Fil., varia por *Insert./Del.* continuos), Grado Rel. (nº Col. ver Unión Compatible (*Alg. Rel.*))

34.6 Claves y relación de tablas

- **Key** : tipos de Claves o Llaves:
 - **ClCand** (Candidata): conj. min. (*TAI*) de Atrib. que Id. de forma *Unic.* cada *Tupla*. Puede haber varias ClCand.
 - **ClPr** : ClCand elegida para Id. las Tuplas. *Primo* (uno de los A. de la ClPr). Ej: DNI, ISBN (libro), i.e. puede estar compuesta por varios Atrib. Cumple las *Restr.* de *Unic.* de Col., Fil. y R. Oblig. (ver).
 - **ClAj** (Ajena o Foreign): Atrib. de una Tabla que coincide con la *ClPr* de otra, i.e. cumple la Restr. de R. Coincid. (ver) pero previamente deben haberse *Rel. Tab.*. No cumple Restric. *Unic.* o *Not Null*. (puede tener valores Null o repetidos). Diferentes polit. de *Actual.*: *CASCADE (SQL)*, etc. Similar a *Punt.* y *Ln*

Ver *Pal. Reserv.*, *VSAM*, *FAT*, *Etiq.*, *Hash*, *Descr. Arch.*, *Arbol B+*, *Malw*. *KeyLogger*, *Direc.*, *ClPr*, *ClAj*, *Ind.*, *Campo*, *Dic.*, *GET PHP*, *Autent.*, *Cript.* *Simetr.*

- **Rel. Tab.** : solo hay 3 *Card.* (Fig. 34.14): Ej: *BD* el clásico **Proveedor-Pieza-Envio** (Supplier, Part and Shipment) de *B*. [Date04] (*Hist. BD*) o Doctor-Paciente-Cita:

- 1:M (o M:1) o uno a muchos o *EpiMorf.* (o *MonoMorf.*): Ej: paciente-cita. 1 paciente hace de (0,n) citas, 1 cita es hecha por 1 paciente
- N:M o muchos a muchos o no *F.* o *Correspond.* No Unívoca: Ej: paciente-doctor. 1 paciente visita (0,n) doctoroes, 1 doctor es visitado por (1,n) pacientes.
- 1:1 o uno a uno o Biyect: para partir partir o *Proyec.* tablas (*Alg. Rel.*)

Es *Vincul.* (extender explicación). Ver *ASSERTION* (abajo). Ver *Join*, *DDL*, *Tabla*, *Restr.* Coincid.

34.7 Normalización de tablas

- **FN** (Forma Normal): partir gran tabla en pequeñas (*Axiom. Ortog., Cohes.*)¹ Para:
 - a) Reducir *Redund.*
 - b) Aumentar *Integr.* (ver abajo)Al partir tenemos una *Jerarq.* padres e hijos. Cuanto más alta sea la FN menos *Vulner.* será a *inConsist. Log.*. Se aplica en *Dis.* tras el paso *ER2Rel* con los obj. de: -) Dismin. problemas de *Actual..* -) *Opt. Consult.* por crear *Ind.* más rápido
Fué iniciado por *Codd Hist. BD*
 - 1 FN o *Atom.*: $\forall Dom_i$ tiene valores atómicos. Esto no ocurre en *BDOO Obj. Rel.*
 - 2 FN: 1 FN y además $\forall ANCP = f(ClPr)$ todo *Atrib.* (que No es de *ClPr*) es completamente *Depend.* de la *ClPr*, i.e. hay *Corr.*
 - 3 FN: 2 FN y además $\forall ANCP \neq ft(ClPr)$ todo *ANCP* no *Depend.* **Transit.**ivamente de la *ClPr*, i.e. $\nexists Corr.$ entre *Atrib.* secundarios.
 - FN Boyce-Codd (BCNF o 3.5NF): es una versión ligeramente más fuerte que 3 FN. 3 FN y además $\forall ClCand_i \cap ClCand_j = .$ para todo par de *ClCand* compuestas no coinciden sus *Atrib.*. *Hist. BD.*

34.8 Restricciones a los datos

- **Restr.** de *Integr.* (Constraints, *Req.*, Ligadura *Fisic.*, *Condic.*): parte del *Esquema* y restringe la:
 - a) *Sintax.* de Datos (*Tip.*) y
 - b) *Semant.* (*Rel. Tab.*).Dan Correctitud y *Consist./Complet.* (lo opuesto es *Corrupt.* (*Perd.*)). Hay 4:
 - R. *Ent.*: R. *Unic.* y *Oblig.* en *ClPr*

¹*Estr.* mejor la informac. (poniendo atributos relacionados **directamente** en una misma tabla y los **indirectamente** en otras manteniendo unos enlace) que enlazan a otros.

- Referencial: R. Coincid. entre ClA_j y su ClPr correspond.
 - *Domin.*: cada atrib. pertenece a un conj. bien def.
 - *Usr.*: resto de R. definidas por el libremente (las de *CASCADE..* del *CREATE* y *Trigger*)
- RESUMEN DE LO QUE SIGUE: explica en detalle los elementos que componen las R. anteriores, que básicamente es las opciones de *CREATE SQL* (Null, CASCADE,..)
 - Detalles de la anterior:
 -
 - R. *Unic.* Col. o Fil. (*UNIQUE* o *EXCLUSIVE*): $\nexists V_i = V_j$ o $\forall V_i \neq V_j$ (Log.1) (ej: *Integr.* Ent.). Ver *Correspond.*, *SQL*, *Truco Combin.*, *MultiConj.*, Esp. Nombres (*Ambito*) *F*, *IP*, *Id*.
 - R. Oblig. (*NOT NULL*): no pueden dejarse valores sin poner o con *Null*
 - R. Coincid. o Referencial: de valores de *Domin.* (ej: *Integr.* Ref.) *IdFinca = IdFinquilla* Fig. 34.14 *Rel. Tab.*, ClA_j puede llamarse de forma distinta pero tiene el mismo Domin. que en Tab. original).
 - R. *Atom.* (1 *FN*): cada Atrib. solo toma un valor del *Domin.*
 -
 - R. Ord. de Tupla y Atrib. no es significativo.
 - R. *Op. BD*: como muestra la Fig. 34.14.
 - Básicas: a) Ninguna acción. b) Transm. en Cascada. c) Poner *Null*. d) Poner Sentencia/Por Defecto.
 - Op. Rechazo: a) Simple: (ej: borrar Tupl. o ClPr no permitido). b) *CHECK* (en toda *Actual.* comprueba si *Predic.* cierto, si no rechaza). c) *ASSERTION*: generaliza *CHECK* pero sobre distintas Rel. (*Rel. Tab.*)
 - Op. No Rechazo: lanzar un *Trigger* (\in *Proced. Almac.*) si se incumple *Restr.*

34.9 Teorema de Codd. Declarativo vs procedural

- Tma *Codd* (*Turing A., Hist. BD*): Dem. que los 2 siguientes L. para *Consult.* (*DML*) tienen el mismo Poder *Expr.*, i.e. *Equiv.* a Log.1, i.e. son *Rel. Complet.*:
 - *Calc. Rel.*: es *Declar.* (ver *Alg. Rel.*) e incluye C. *Tupla* y C. *Domin.* (ver abajo)
 - *Alg. Rel.* (*Proced.* ver)

34.10 Calculo relacional

- *Calc. Rel.* (no confundir con C. *Constr.*): parte del *Mod. Rel.* para hacer *Consult.* de forma *Decl.* a diferencia del *Alg. Rel.* (que es *Imper.* [*CalcRel*]), ambos *Equiv.* por Tma *Codd*. Requiere *Opt. Consult..*
Dos subtipos:

- C. R. *Tuplas* (CRT): forma parte del *Mod. Rel.* basado en *Busq.* de T. que cumplan cierto *Predic.* (=Topos). Es el calc. original propuesto por *Codd Hist. BD* que inspiró *QUEL* y *SQL*.
- C. R. *Domin.*: usa símbolos de *Log.1* (\forall, \exists, \dots), de *Conj.* (\wedge, \vee, \dots) y *Op.* CRT. Propuesto por [Lacroix&Pirotte] *Hist. BD*. Creo es *Decl. Log.*

Ver C. *Lamb.*, C. *Pi*, C. *Zuse IEEE 754*

34.11 Operaciones y lenguajes del cálculo relacional

- *DML, DDL, DCL*
- *Op. BD: Actual., Consult. (Select.)*
- *Persist. Proced. Almac.*
- *Opt. Consult.: Ind. y Factor de Block*

34.12 Algebra relacional

- *Alg. Rel.* : es uno de los 2 tipos de *DML*. Creado por *Codd* y es *Proced. cambios Est. Sindic.* con Op. Primit.
 - Es la **Teoría Fundacional** de L. *Consult.* como *SQL* basado en *EA* y *Semant.* Bien Fundada (*Log.*). Basada en *Op. Primit.* (*Union,..*) y Derivadas.
 - El I. es una *Rel.* y O. otra Rel. Ej: $Rel_c = \Pi(Rel_a \times Rel_b)$ (*Prod. Cart.*)
 - *Alg. Rel.* es *Imper.* [CalcRel, Inglés] o *Proced.* [CalcRel, Esp] porque las op. se ejec. de izda-dcha y de dentro-fuera *Recurs.* [CalcRel, Ing].
 - *Calc. Rel.* es *Decl.* (o *Predic.* o No Proced.) porque los cambios de *Est.* son especificados con *Predic.* que definen el Est. Objetivo pero sin *Sindic.* las op. para llegar a el.
- IMPORTANTE: por esto necesario un *Opt. Consult.*

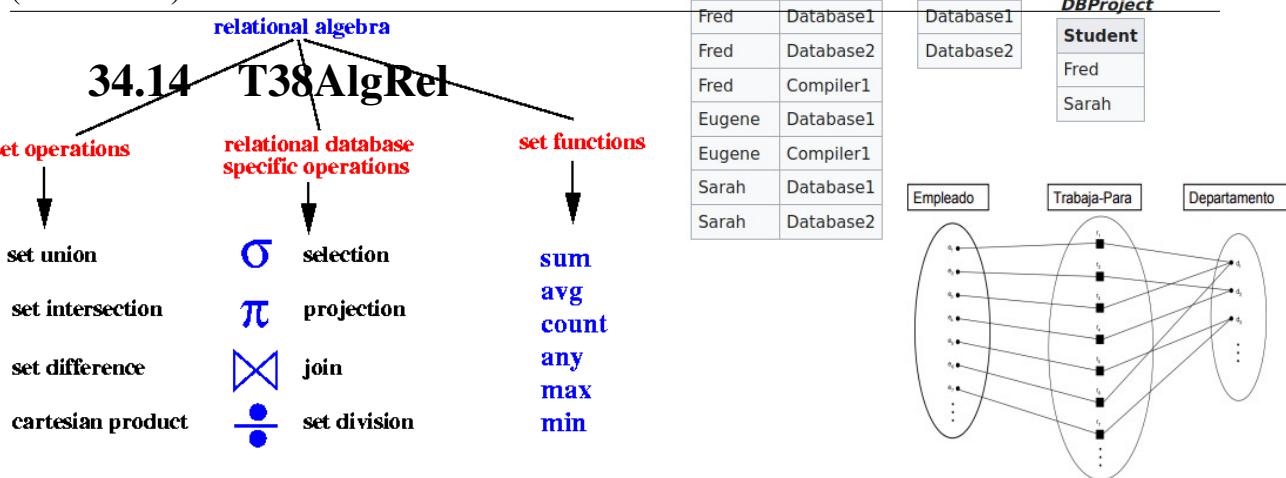
¿Como pasaria un *Opt. Consult.* una Consult. SQL a Alg. Rel.? Ej: *SELEC a FROM T1 INNER JOIN T2 AS T1.c=T2.c WHERE b>30* → F. de Proyec. y Joins..

34.13 Operaciones del álgebra relacional

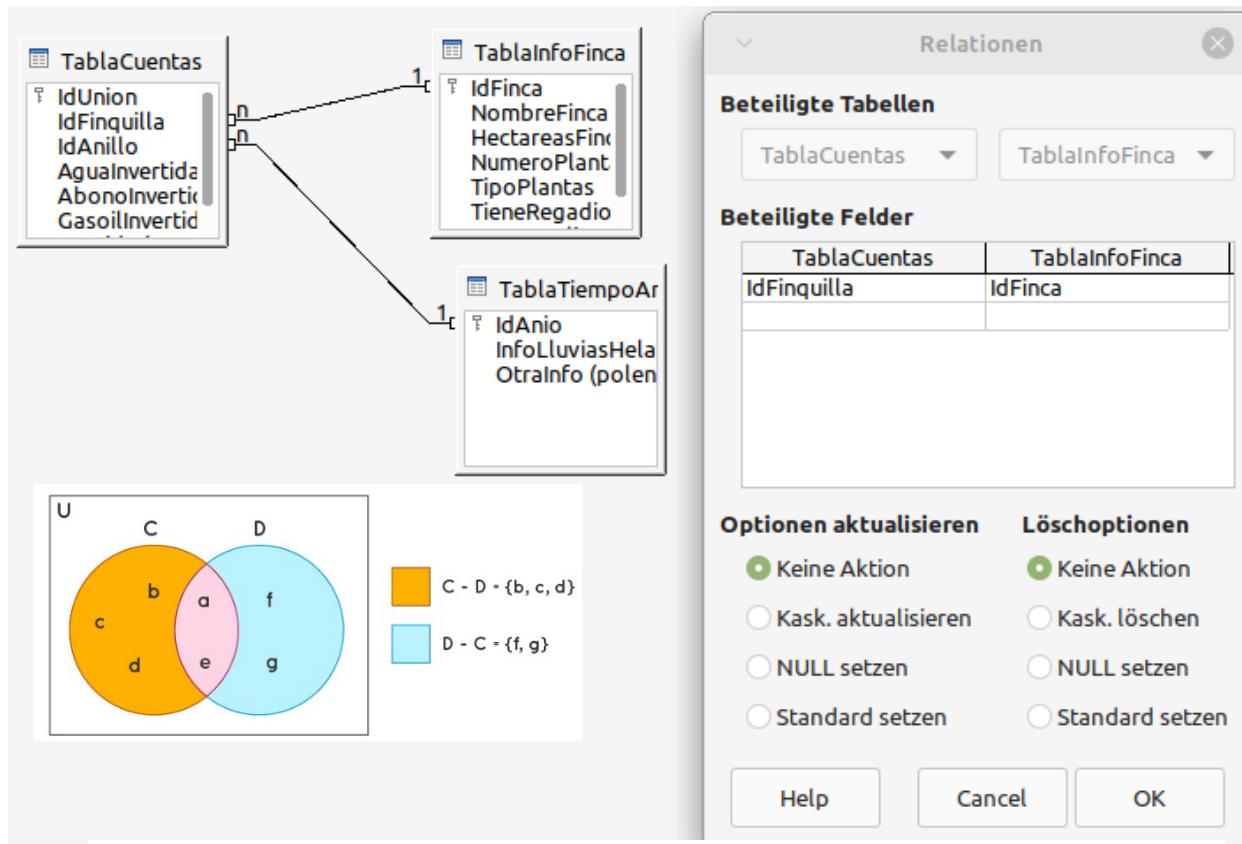
- Según I. de Op.: UnAria o BinAria.
- *Op. Asign.*: almacena en una *Rel.* el resultado de un *Consult.* sobre otra *Rel..* Permite: a) *Vincul.* (denominar resultados intermedios de *Div.* *Venc.* op. complejas) b) Copiar (una *Rel.* en otra). c) Renombrar *Atrib.*
- *Op. Primitivas* (ver *SQL*):

- Selección: $\sigma_{\text{AguaInvertida} > 1000 \wedge \text{CantidadCosecha} > 500}(\text{TablaCuentas})$.
- Proyec.: $\Pi_{\text{AbonoInvertido}, \text{GasolinaInvertida}}(\text{TablaCuentas})$. Extrae las Column. (Atrib.) indicadas. Ver *DISTINCT SQL, Rel. Tab. Card. 1:1*.
- **Prod. Cart.** (Prod.): $\text{TablaInfoFinca}_{f_1, c_2} \times \text{TablaTiempoAnual}_{f_2, c_2} \approx \text{TablaCuentas}_{f_1 * f_2, c_1 + c_2}$.
 El I. son varios *Conj.* o *Domin.* y el O. es el *Conj.* de *Tuplas* de todas las *Combin.* posibles: $A \times B = \{(a, b) : a \in A, b \in B\}$.
 Equiv. a un Cross Join. Ver *Rel., Pullback, SQL, Truco Combin.*
- Unión: $R \cup S$ donde ambas deben ser **Compatibles en Unión** i.e. $\{\text{Atrib}(R)\} = \{\text{Atrib}(S)\}$, i.e. ambas Rel. (o Tablas), tener el mismo conj. de Atrib. o mismo Grado
- Diferencia: $R - S = T$ ambas deben ser Compatibles en Unión. $T = \{t : t \in R \wedge t \notin S\}$ contiene las *Tupla* que están en *R* pero no en *S*, ver Fig. 34.14
- *Op.* Derivadas:
 - **Join** (Reunion) o *Pullback* de *Cat.* o Prod. *Fibrado*: $R \bowtie S = T$. Donde *T* es la Tabla Extendida/Expandida/Subst. de la Rel. Tab. (con las *Tuplas* para las que hay *Vincul./Enl.* completos).
 Es la *Compo.* de *Rel.* (no confundir con *Vista* que es de *Consult.*).
 Se puede derivar de op. simples como el *Prod. Cart.* y la *Proyec..*
 Este es el Join Natural (o inner que se puede hacer con *WHERE* ver *SQL*) pero hay otros (left, right and cross=Prod. Cart.) donde si faltan datos se ponen *Null* Fig. 34.14. Ver *Truco Combin.*, J. Factor (*Opt. Consult.*)
 - Intersección: $R \cap S = R - (R - S)$ ambas Compatibles Unión. Tuplas que están en ambas
 - División: $R(x, y)/S(y) = T(x)$ con $T(x)$ Tuplas $R(x)$ para los que hay valor en $S(y)$ ver Fig. 34.14.
 Es un tipo de *Join*

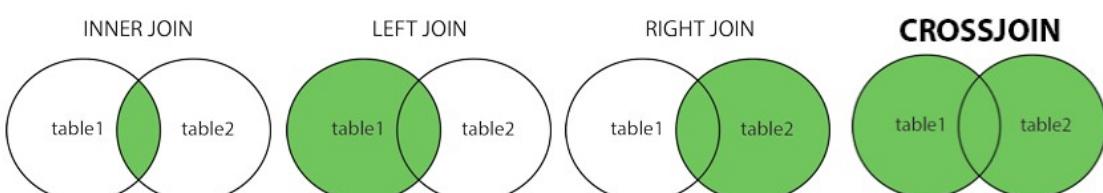
34 – +38. Modelo de datos relacional. Estructuras. Operaciones Algebricas Relacionales
(10NovFuen)



Employee			Dept		Employee \bowtie Dept			
Name	EmplId	DeptName	DeptName	Manager	Name	EmplId	DeptName	Manager
Harry	3415	Finance	Finance	George	Harry	3415	Finance	George
Sally	2241	Sales	Sales	Harriet	Sally	2241	Sales	Harriet
George	3401	Finance	Production	Charles	George	3401	Finance	George
Harriet	2202	Sales			Harriet	2202	Sales	Harriet
Mary	1257	Human Resources						



- INNER JOIN: Returns records that have matching values in both tables
- LEFT JOIN: Returns all records from the left table, and the matched records from the right table
- RIGHT JOIN: Returns all records from the right table, and the matched records from the left table
- CROSS JOIN: Returns all records from both tables



Chapter 35

+*39. Lenguajes para la definición y manipulación de datos en sistemas de base de datos relacionales. Tipos. Características. Lenguaje SQL. (18FebJoaq)

35.1 Introducción. Conclusión

- Ej. Motivador: ¿Lenguaje *Complet.* para consultas rápido?
- *Hist. BD:* Frege *Log.* 1, Codd70 *Calc. Rel.*, [Lacroix&Pirotte], Ingress, Boyce *IBM72 (SQL)*,
- Futuro: *NoSQL* Corpus para *Aprend. Autom.* y no hacer *Consult. SQL*

35.2 Bases de datos relacionales

- *Mod. Rel.* (Fig. con *Rel. Tab.*)
- Jerarq. de SubL. *SGBD: DDL, DCL..*
- *Arq. ANSI/SPARC:* Fig. 35.8 con *Vista Interna, Conceptual y Externa.*

35.3 Lenguaje de definición de datos

- *DDL* (Dat. Def. L.): subL. de *SGBD* que permite Def. el *Esquema*.
 - Comand. principales: *CREATE, ALTER, DROP, FOREIGN* para def., modif., borrar y *Rel. Tab.*
 - *Esquema BD:* def. con *DDL*.

- **IMPORTANTE:** Creo se encarga solo del Nivel Conceptual de *Arq. ANSI/SPARC*. Aunque F. Javier dice permite especificar los 3??.

Ver *Rel. Tab., Descr., Dic. BD*

35.4 Lenguaje de control de datos

- **DCL** (Dat. Control L.): subL. de *SGBD* que permite al *Admin.* modificar el *Permiso* de Acces. a Dat. y a Comandos en entornos Multi*Usr* que comparten Dat. *Concurr.*. Actua de *Interf.* de Usr y Admin. (en *Arq. ANSI/SPARC??*)
 - Comand. principales: **GRANT,REVOKE** para quitar y denegar Permiso a *CONNECT, SELECT, Insert....*

Ej:

```
GRANT SELECT, INSERT, UPDATE, DELETE ON DB1.Employee TO user1;
REVOKE SELECT, INSERT, UPDATE, DELETE ON DB1.Employee FROM user1;
```

Hay quien los pone con *DDL* y *DML* y relacionado con *TCL Transac.*

35.5 Lenguaje de manipulación de datos. Teorema de Codd

- **DML** (Dat. Manipulation L.): subL. de *SGBD* que permiten las 2 **Op. BD** básicas (como *Estr. Dat. Dinam. (Lista)* o *Fich. (SAM)*):
 - *Actual.:* o Update: *Insert./Borrar/Modif.* Ej: **INSERT, SELECT-FROM** ambos de *SQL*.
 - **Consult.** (Query): mostrar una serie de *Atrib.* (Col.) y dentro de ellas, una serie de *Tuplas*, que cumplan una *Condic.* Puede ser **SELECT**. (sobre ciertas *Tablas*) o total (a toda la BD). Equivale a *Eval. Predic.* (=Topos de Busq.=Asoc.=Recuperar=Retrieve=subconj. dat. condic). Es *Unif.* por *Busq. Vel.* (ver *Ind.. Ver DQL (SQL), jQuery (JavaScript)*)

Ver *Transac., Restr. BD, Trigger, Informe*

- Tma *Codd*: los 2 tipos (Alg. Rel. vs Calc. Rel.). L. Comerciales: Calc. Rel. Tuplas, Declar. y amigables Ej: *SQL, QUEL, y QBE*

35.5.1 Lenguajes comerciales (aparte de SQL): QUEL, QBE y FQL

- **QUEL** (Query L.): *DML* basado en *Calc. Rel. Tuplas* (ver):
 - Ingress: *Hist. BD* es la implementación *SGBD* de *QUEL* de Berkeley que en los 80 se consideró el L. más cercano al *Alg. Rel.* de *Codd* sobre todo en lo referente a *Composabilidad*, pero fué reemplazado por *SQL* al ser menos matemático y sencillo de entender. Tiene 2 formas de trabajo: modo interactivo y modo inmerso.??
 - PostgressSQL: hoy es la continuación de Ingress. Es *BDOO Obj. Rel. Libre*

- XQuery: *Consult.* sobre *XML* o *JSON*
- *QBE* (Query By Example): desarrollado en *IBM75* en paralelo a *SQL*. Fue el 1º *DML* con *GUI* e inspiró al resto como *Access Hist. BD*. Las *Consult.* se hacen mediante *Tabla Esqueleto* y Ejemplos (Fig. 35.8). Las *Op. BD* se hacen con los *Comandos*: *P. D. I. U.*
Ej: la 1º fila se repitiría junto a las otras

Descrip.	NombreTabla	campo1	campo2	campo3	campo4
<i>Consult.:</i> mostrar valores de <i>c1</i> y <i>c3</i> cuando <i>c2 > 16</i> y <i>c4 = Cadiz</i>		P.valor1	>16	P.valor3	Cadiz
Delete: por consulta, solo D. 1 a 1	D.			valor	
<i>Insert.:</i> idem ant.	I.	valor1	valor2	valor3	valor4
<i>Actual.:</i> idem ant.	U.	criterSelec.		criterSelec	

- *FQL* (Functorial Query L.) [Spivak]: basado en *Cat..* ver *Migr.*

35.6 SQL

- *SQL* (Structured (*Estr.*) Query Language): L. *Decl.* (4GL) y de *Domin.* Específico para Progr. Como en *SGBD* se diferencian 3 subL.: *DDL*, *DML* y *DCL*. Por [Chamberlin y Boyce] *IBM72 (Hist. BD)*, *Actual.* SQL2016 por *ANSI/ISO*. Se considera el L. *Estand.* de Facto integrado en cualquier *SGBD* (*MySQL*, *Access..*), se puede integrar en *PHP*.
 - No *Sens.* al caso: da igual may. que min. pero *Comandos* may., *Tablas* 1ª may., Col. (*Atrib.*) y resto minusc.
 - *Separ.* sentencias con ;

Ver *NoSQL*, *MySQL*, *Opt. Consult.*

35.6.1 DDL: CREATE

- Crear *Tabla*:

Sintax. simple:

```
CREATE TABLE NombreTabla (
    col1 TIPO,
    col2 TIPO,
    PRIMARY KEY (col2 , col4 ,...)
);
```

Ej:

```
CREATE TABLE Usuario (
    carnet VARCHAR(10) PRIMARY KEY,
    nombre VARCHAR(8),
    direccion VARCHAR(8),
    fecha_ingreso DATE NOT NULL
);
```

Sint. compleja: con *Restr. CASCADE...*

```
CREATE TABLE NombreTabla (
    col1 TIPO [NOT NULL] ,...
    [CONSTRAINT nombreClave [UNIQUE | PRIMARY KEY | NOT NULL]
     (colClave)]
    [CONSTRAINT nombreClaveForanea FOREIGN KEY (columnas) REFERENCES
     Tabla (columnas) [ON DELETE|UPDATE] [SET DEFAULT|CASCADE|
     SET NULL] ]
    [CONSTRAINT nombreCondic CHECK (condic.)]
);
```

Ej: El CHECK se puede reemplazar por un *Trigger*

```
CREATE TABLE Trabajos (
    mat VARCHAR(8),
    dni VARCHAR(9),
    horas NUMBER(3,1) CONSTRAINT
        ck_trabajo CHECK (horas >=0.5),
    CONSTRAINT fk1_trabajos FOREIGN KEY (mat)
        REFERENCES Coches,
    fecha_rep DATE
);
```

Descripción detallada:

- *Tip.* Dat.: *CHAR, NUMBER, INTEGER, DATE, LONG, VARCHAR??*
- Opciones: *Not/Null* (no admite valores nulos), *Unic.* (no valores repetidos no *MultiConj.*)
- *ClPr*
- *ClAj* con opc.:
 - * *ON UPDATE/DELETE CASCADE*: si se modif./elim. *ClPr* correspondiente, esta también se modif./elimin. (como *Ln Duro*)
 - * = *SET NULL*: =, esta se pone a *Null*
 - * = *SET DEFAULT*:=, = a defecto

35.6.2 DDL: ALTER/TRUNCATE/DROP TABLE

- Modif. *Tabla* Sint.:

ALTER TABLE NombreTabla

```
ADD nombreCol TIPO | MODIFY nombreCol TIPO | DROP nombreCol |
ADD CONSTRAINT restric. | DROP CONSTRAINT restric.
```

Ej1:

```
ALTER TABLE Coches ADD modelo VARCHAR(15);
```

Ej2:

```
ALTER TABLE Trabajos ADD CONSTRAINT fk2_trabajos FOREIGN KEY (dni)
    REFERENCES Mecanicos;
```

- Eliminar:

TRUNCATE/DROP TABLE NombreTabla ;

ambos eliminan todas las filas pero tabla si/no permanece.

35.6.3 DML: INSERT INTO, DELETE FROM, UPDATE

- *Insert.* Actual.: en Pila/Cola es Append. Es Unif.. Ver Estr. Dat., Dinam., Lista, Arbol, Hash, Embeb.

INSERT INTO Tabla (campo1 , . . . , campoN) **VALUES** (valor1 , . . . , valorN)

DELETE FROM Tabla **WHERE** condic .

UPDATE Tabla **SET** campo1=valor1 , . . . , campoN=valor **WHERE** condic .

Ej:

```
INSERT INTO Clase VALUES(3,365);
```

Ej: Copiar de una tabla Origen (SupplierName) a otra Destino (Customers)

```
INSERT INTO Customers (CustomerName, City, Country)
SELECT SupplierName, City, Country FROM Suppliers;
```

35.6.4 DDL?: VIEW, CREATE INDEX

- *Vista* (no confundir con *SELECT INTO*): *Tabla Virt.* o SubConj. Dat. resultado de una *Consult.* usando varias Tab. o Vistas (*Recurs.*). Ocupan menos *Mem.* pues no *Almac.* los dat. sino la def. Vista. Cuando *Actual.* una tabla cualquier Consult. que use la Vista cambiará *Autom.* (comprobado en LibreOffice Base). Es como una *Var.* que *Almac.* resultado intermedio. Es *Unif.* pues permite *Compo.*ner Consult. (no confundir con *Join*).

Sintax. Crear/eliminar V.:

CREATE VIEW NombreVista **AS** subconsulta ;

DROP VIEW NombreVista ;

Ej:

```
CREATE VIEW [Brazil Customers] AS
  SELECT CustomerName, ContactName
  FROM Customers
  WHERE Country = 'Brazil';
```

- *Ind.*: *Enum.* para *Opt. Consult.* (*Busq. Vel.*) Sint. Crear/eliminar V.:

CREATE INDEX NombreInd **ON** NombreTabla (col1 , col2 , . . .) ;

DROP INDEX NombreInd

35.7 DQL: SELECT

- *Consult.* Fig. 35.8

Sintax.

```
SELECT [ALL | DISTINCT | DISTINCTROW] listaAtrib1 | *
[INTO cursor] FROM ListaTablas
[WHERE cond1]
[GROUP BY listaAtrib2]
[HAVING cond2]
[ORDERED BY listaAtrib3 [ASC|DESC], ...];
```

Ej: “Obtener el cifc de todos los concesionarios cuya cantidad en tabla Distribucion esté entre 10 y 18”

```
SELECT DISTINCT cifc FROM Distribucion
WHERE cantidad >=10 AND cantidad <=18;
```

Ej:

```
SELECT cifc , ROUND(AVG(cantidad),4)
FROM Distribucion
GROUP BY cifc HAVING AVG(cantidad)<=10;
```

Ej: Inner Join se puede hacer con WHERE

Ver ejerc. INNER JOIN R ON A.A1=B.B1 o con WHERE A.A1=B.B1

Descripción detallada:

- listaAtrib.1: a los que se aplica Proyec.
- DISTINCT: La Proyec. por defecto da un MultiConj. (no Unic., con duplicidades de la propia tabla o del Prod. Cart.) y D. lo evita. ALL muestra todo. Ver Truco Combin.
- FROM: tablas que se usarán. AS: alias para nombre de alguna Col. de SELECT. Fig. 35.8
- WHERE: selec. de Tuplas que cumplan una Condic. o Predic. sobre Atrib.. Rel. ClPr con ClAjs. AND: une Condic.
- GROUP BY: cuando una Col. es MultiConj. podemos agruparlas si las Col. mostradas se agrupan igual. Contrario a DISTINCT.
 - * HAVING: cond. que le acompaña es la misma que la que acompaña a SELECT, restringe aún más los grupos formados por GROUP BY (se usan con Op. Agregac.)
- ORDERED BY (columna/alias) DESC: ordena (última) Col. según criterio. LIMIT (2): muestra solo las (2) primeras filas de un ordenamiento. Fig. 35.8
- Op. Agregación (SUM|MIN|MAX|AVG): en HAVING o en Atrib.
- Op. para Expr. (ver Alg. Rel.): como todo L.: - Comparar (<>, <=, !=), - Comparar Cadenas: LIKE (es como == en C) - Op. Log. AND, OR, NOT, y Op. Conj. UNION, INTERSECT, MINUS . - F: numer. alfanum. conv. datos
- Otros: ANY, EXISTS, IN

Ej: “P10.10: Obtener las parejas de modelos de coches cuyo nombre es el mismo y cuya marca es de Madrid”. Truco Combin.: $i < j < k$ (del Princ. Inclus.-Excl.) reduce Combin. repetidas del Prod. Cart.. Usar en Join???. Ej: A.modelo>B.modelo

```
SELECT A.modelo , B.modelo
FROM Coche AS A, Coche AS B, Marco , Marca
WHERE A.nombre=B.nombre AND A.modelo>B.modelo
AND A.codcoche=Marco.codcoche AND Marco.cifm=Marca.cifm AND Marca.ciudad='Madrid';
```

35.7.1 DQL??: SELECT INTO, AVG-GROUP, COUNT

- *INTO* (no confundir con *VIEW Vista*): crea una especie de *Var.* o *Tabla Tmp* copiandole las col. deseadas. Sirve para *Backup* de tablas

Ej:

```
SELECT * INTO CustomerGermany
  FROM Customer
 WHERE country='Germany';
```

Ej: *Backup*

```
SELECT CustomerName, ContactName INTO CustomersBackup2017
  FROM Customers;
```

- Promedio

```
SELECT cife , AVG(cantidad)
  FROM Distribucion
 GROUP BY cife;
```

- Contar

```
SELECT COUNT(DISTINCT nombre)
  FROM Marca
 WHERE ciudad='Madrid';
```

35.7.2 Procedimientos de Almacenamiento: PROCEDURE

- *SQL/PSM (Persist. Stored Modul., derivado de PL/SQL): Exten. def. por la ISO para SQL. Es un L. Proced. para usarse en Proced. Almac..*

– *Proced. Almac.* : *Subrut.* que responde a *Petic.* de Apl. de *Usr de Acces.* a BD. Su ventaja es que se *Almac.* en el *Dic. BD* y se ejecuta en el mismo Motor BD (*SGBD*) evitando tener que transm. todos los datos al *Usr*. Ej:

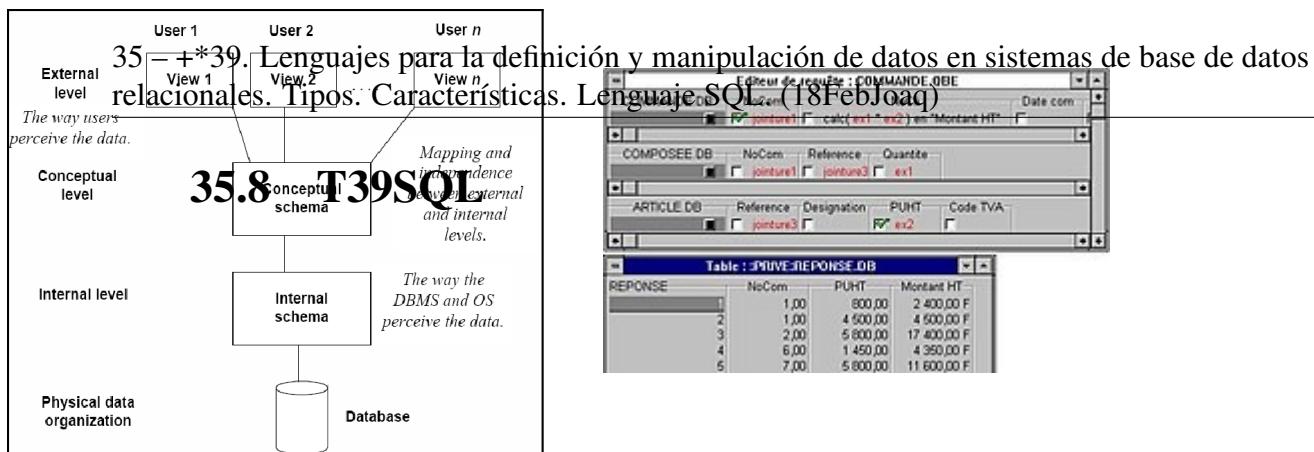
- a) Lanzar *Trigger* si se incumple *Restr.*, también se puede hacer con *CHECK*
- b) Quitar automáticamente un valor de TablaStock (TablaProductosEnAlmacen) cuando añado un valor en TablaVentas (porque un cliente ha comprado) Ver *Event.*
- c) Hacer *Persist.*: guardar en un Hist. cambios (ver ej. *Trigger* abajo)

– *Trigger* ∈ *Proced. Almac.* : *Subrut.* o *Proced.* lanzada cuando no se cumpla una *Restr. Semant. (Integr.)* para mantener *Integr.* y hacer *Persist.* ver ej. abajo. Ver *Transac., Event..*

```
CREATE PROCEDURE nombreProcedimiento([ parametro1 , parametro2 , . . . . . ])
[ Atributos ]
BEGIN Instrucciones
END
```

Ej. *Persist.* con *Trigger*

```
CREATE TRIGGER trigger_prod_hist
..... Ver Libreta Javier ....
```



The screenshot shows the following:

- Ecran de requête : COMMANDE.QBE**: A query builder window with fields like 'jointure1', 'calc(ex1 * ex2) = "Montant HT"', and 'Date com'.
- Table : 3MOVE.REPONSE.DB**: A data table with columns 'REPONSE', 'NoCom', 'PUHT', and 'Montant HT'. The data is as follows:

REPONSE	NoCom	PUHT	Montant HT
1	1,00	800,00	2 400,00 F
2	1,00	4 500,00	4 500,00 F
3	2,00	5 800,00	17 400,00 F
4	6,00	1 450,00	4 350,00 F
5	7,00	5 800,00	11 600,00 F

```

SELECT "TablaTiempoAnual"."IdAnio", "TablaInfoFinca"."NombreFinca",
"TablaCuentas"."CantidadCosecha", "TablaInfoFinca"."HectareasFinca",
"TablaCuentas"."CantidadCosecha" / "TablaInfoFinca"."HectareasFinca" AS
"Rendimiento" FROM "TablaCuentas", "TablaInfoFinca", "TablaTiempoAnual"
WHERE "TablaCuentas"."IdFinquilla" = "TablaInfoFinca"."IdFinca" AND
"TablaCuentas"."IdAnillo" = "TablaTiempoAnual"."IdAnio" ORDER BY
"Rendimiento" DESC LIMIT 2

```

The screenshot shows the query definition in a table-based interface:

Feld	IdAnio	NombreFinca	CantidadCosecha	HectareasFinca	"TablaCuentas"."CantidadCosecha" / "TablaInfoFinca"."HectareasFinca"
Alias					Rendimiento
Tabelle	TablaTiempoAnual	TablainfoFinca	TablaCuentas	TablainfoFinca	
Sortierung					absteigend
Sichtbar	<input checked="" type="checkbox"/>				
Funktion					
Kriterium					
oder					

Below the interface, the resulting data table is shown:

IdAnio	NombreFinca	CantidadCosecha	HectareasFinca	Rendimiento
2020	VegaMotril	63456	40,00	1586,4
2019	VegaMotril	52345	40,00	1308,62

Chapter 36

41. Utilidades de los sistemas gestores de base de datos para el desarrollo de aplicaciones. Tipos. Características.

- RESUMEN ver TSGBD
- *Migr.* : Export/Import/Fundir *Esquema* de un SGBD a otro. El *FQL* lo permite mediante Δ , Σ y Π (*Pullback*, P_i). Es una forma de *Preserv. Digit.*. Ver *Hash*
- *JSE* (Java Standard Edition): es cerrada y de *Oracle*. Para *Desarr.* y *Despl.* de Apl. Desktop y Servid.. Tiene al JEE (*Progr. Compo.*) y al JDBC (*Conect. BD*).
- *Conect. BD* : API o *Conect.* para hacer *Llam.* de Acces. a un *SGBD* desde una Apl. (L.). Es un *Driver*.
 - ODBC: de *Microsoft*
 - JDBC: del *JSE*
 - Otros: ADO (ActiveX, *Progr. Compo.*)
- *Progr. Compo.* (Ing. Softw. basada en *Compo.nentes*): casi *Parad. Progr.* que permite *IPC* (comunicar progr. en Red.) y relacionado con *Persist.* (ver abajo Java)
 - **Obj. Distrib.**: son *Obj.* de *POO* pero donde el *Pas. Mens.* no solo es posible entre *Obj.* del mismo Progr. o SO si no entre distintos *Comput.* via *Red* y de forma *Transpar.* (parecen locales). Ej: **CORBA** (Common Object Request Broker Architecture) es un *Estand.* OMG para facilitar la *Interop.* de *Petic..* Gracias a *QNX*.
 - COM (Component Object Model): reemplazado por Ej: **ActiveX** (no confundir con ReactivX *Event.* ni DirectX (*Render:*)), se descarga como los *Applet* de *Java* pero limitados a ejec. solo en *Naveg.* de *Win* (no en *JVM*, no *Portab.*), se carga rápido el *Obj.* porque se guarda *Persist.* en *MS*. Producen confusión al descargar un *Malw..* Por todo esto dejaron de usarse *Hist. R..* Ver ADO *Conect. BD*.
 - **Jakarta Enterprise Edition** (JEE): *Exten. Open-Libre* del *JSE* Ej: Jakarta Servlet: para Servid. Apl. (ver *Applet*). Ej: Jakarta Enterprise Beans (EJB \in JEE): permite *Persist.* (y *Transac.* *Segur.* JTA) gracias a su *Integr.* con el **Jakarta Persistence API (JPA)** para *Map.* *Obj.* *Rel.* (*BDOO*) que *Unif.* el Java Data Object (JDO) y el EJB3 *Container*. Creo comunica datos con *XML* o *JSON*.

- Otros: React (*JavaScript*).

Ver *BDOO*, *Applet de Java*, *IPC*

- *SQL/PSM* derivado del *PL/SQL*
- *PhpMyAdmin*

Chapter 37

Común Ingeniería del Software

- *Ing.* Softw. \subset Resol. Probl.: un Programador lo es pero sin educación Ingeniera. Ver Arq. Softw.
 - I. Inv.: averiguar como funciona un Micro o Progr.. Ver Depur., NTFS (Sist. Arch. Win), Libre (Open Source)

Ver Desarr., Proyec., Creat., I. Social (Confid.)

- ReFact. : mejorar un código (en fase Manten. en Efic. y Eleg.) sin modificar su comportamiento Semant. (sin que lo note el usr Transpar.). Gracias a Modul. y Encapsul.. Ej: Concurr.. Ver Part., DesCompo.. F. Primos
- Depend. : Ver Infierno de las D. (Acopl., Modul.), Bibl., 2FN, Vers. (Git), MDI, RAID (Redund.), Markov, Prob., Independ. Log.-Fisic. de SGBD, Instal. (apt), Corr.
- Req. uisitos Funcionales (Ligaduras en Fisic., Restr. en BD, Condic.): Lista de tareas del Progr.. Parte del Anal. en Desarr., donde el analista de la Empresa al hablar con un Client. pasa a un Docum. Texto.
Luego se pasaria a un Model. ER o Diagr. UML. Puede ser ambigüo o Redund. y Evol. como Met. Cient..
 - Req. Técnicos: contrastan con los F., dependen del Hardw. o Softw., elegir L. (Python, Java..), SO (Win,...)
 - Prototipo Visual, menús, GUI sin llenar Modul.. Separar parte Visual (la que interacciona con Usr) de parte Log. (ej: BD de lista de personas)
 - Diagr. UML de Casos de Uso: Acciones (circulos), Sist. (rectang.), actores (muñecos). Ej. Fig. 37.6 (de Anaya) muestra los Permisos de Admin. y Usr sobre una BD.

Ver Dis. POO, Agil

37.0.1 Arquitectura del software

- Arq. : en Ing. Softw. es la Descr. de la Estr. de un Sist. (“se dice” Wittg.) que enfatiza la organización Jerarq. de sus Compo. (Hardw.).

- **Capa** o **Nivel** o **Tier**: de *Abstr.* reduce posibles *Compo.* *Combin.* y permite la *Transpar.* Ej: *Sintax.* en C. Ver *Jerarq. Mem., Jerarq., Arbol, Ext. Red, Segur., 3 C. Ethernet.*
- A. Referentes (*Estand. de Facto*): A. *Neumann, Arq. Kernel, Arq. ANSI/SPARC, Pila Protoc.*, A. Sist. *Informat.* (*SO*). A. *Client. Servid.* (*Frontend-Backend*).

Es *Unif..* Ver *Jerarq. Mem., API, Biolog., Equiv. Wolfram, Vista*

37.0.2 Despliegue

- **Impl.** ementar (no confundir con *Implantar=Despl.*, ni *Instal.*): *Proyec.*tar una idea, paso de *Abstr.* a *Fisic..* Es un “se dice” Wittg.. Ej: I. un *Plan.* o un *Alg.* (*Progr.*). La I. puede usar un *L. Estr.* o *POO* o *BD* (ver *Dis.*). Con *CASE* pasamos de *Diagr.* a *Cod..* Es *Unif..* Ver *Instanc., Eval., Tesis Church-Turing, Her., Enl. Din. POO, Mod. Direc., Micro, Firmw., Conmut., TCP/IP, TCP/IP vs OSI, Topos, Estr. Dat., Recipr.* (*Implicar*), *Mod. Rel.*
- **Despl.** Softw. (*Deployment*): actividades que hacen que un softw. esté *Disponib.* para su uso y son: *Public.ar* (*Release, GitHub*), *Instal.* y *Actual..* Parte del *Desarr.* coordinado por *DevOps*. D. suele ser sinónimo de *Impl.antar* (poner softw. en muchos PCs). Ej: *Kubernetes* y no debe confundirse *Instal.* (ponerlo en 1 solo). Tampoco con *Impl.mentar* (*Fisic.alizar*) que puede ser parte del D. Ver *Fibra, BD Nube, Active Dir. (Servic. Dir.), Disco CD-ROM, JSE*

37.0.3 Generalización y patrones

- **Abstr.** : *Gener.* *Alg.* *F.* o *Sist.* (*Fisic.*) para más *Tip.* datos. Es un “mostrado” Wittg.. Ej. una *Cl.* es abstr. pero un *Obj.* es una *Impl.ementación.* *Tip. Dat. Abstr., POO, Reus. Cod..* Ver *Evol. A. (Cooper. Hist. L.)*.
- Progr. **Gener.** : estilo donde los *Alg.* son escritos en términos de *Tip.* especificado más tarde (como arg. de I. en la F.). Es *Filos.* de *Abstr.* Es *Unif..* Ver *Fisic., Probl., Parad. Alg..*
 - Generic en *Python*
 - *Parametric Polimorf.* en *Haskell*
 - Templates en *Cpp* (ver Práctico)

Ver *Probl. Abstr., Patr., Discrim., MultiConj. Hipergraf., Prod.*

- **Reus.** Cod.: *Metr. Softw.* las *Bibl., Patr. Dis., Abstr.* y la elimin. de *Tip.* lo permiten. Ver *Abstr., Her., Polimorf., Transpar., Arq. ANSI/SPARC, MS, Cache*
- **Patr. Dis.:** Libro [Gamma94] *Hist. I.. Gener. Alg.* permitiendo *Reus..* Se pueden representar con *Diagr. UML*. Hay 3 tipos [Betta]:
 - Creacional: crear *Obj.* en complejas *Jerarq.* de *Her..* **Factory**: crear (o eliminar) enemigo y posición en T. Ejec., *Her.* de Cl. Enemigo. que tendrá una lógica compleja en función de posición y entorno [Betta]. **Builder**: crear mapa nivel complejo de Cl. Mapa.

- Estructural: **Composite**: crear muchos pequeños subenemigos (minions) que mueran cuando el principal muera *Recurs.*
- Comportamiento: **Strategy**: desacoplar los comportamientos (tipos de ataque, flechas, ondas,...) de los Personajes (enemigos) para crear variabilidad y mezclas.
- Otros: Decorator
- Otros: Mod. *Vista Controlador*.

Ver Arq., Meijer, Esquema.

37.0.4 DevOps y disponibilidad

- **DevOps** (Development (Programador) Operations (tipo IT)): intenta acortar el ciclo *Desarr.* complementando el D. Agil.. Autom. y Monit. todos los pasos (integración, *Depur.*, infraestructura (*Disponib.*), *Despl.* y *Admin.*). Ver *Kubernetes*
- Alta **Disponib.** (no confundir con Alto *Rend.*): basado en *Redund.* para evitar *Fall.* en *Centro Dat.* (*Instal.* *Segur.*) Ej. otros: duplicar *Cables*. Usa la *Metr.* del 9 (ej. 3 nueves $0.999=99.9\%$, ver abajo).
 - SAI (Sist. Alim. Inint. o UPS Uninterruptible Power Supply): *Electr.* para que funcionen siempre los *Servid. BD*
 - RAID (ver *Redund.*) y *Backups* (ver)
 - Centro de Respaldo (ver *Centro Dat.*)

Ver *Monit.* Red, *Masiv.* Ver CiberSegur., Acces., QoS, Preserv., *BD Nube*, Apache Spark (*BD Distrib.*)

- Tolerancia/Resilencia/**Robust.** a *Fall.* (*Err.*): *Metr.* de *Calidad*. No afecta que un *Nodo* de *Red* caiga. gracias a *Redund..* Ver Topol., *Red Superp.*, Persist., Manten., *Monit.*, RT, *Manej.*

Si Tiempo Medio *Fall.* $MTTF = 81$ años y T. M. Reparación $MTTR = 1$ h. \Rightarrow *Interr.* Por Año = 0.0123 h/año

- **Domin.**
 - D. Active Dir. (*Servic. Dir.*)
 - D. DNS (*ICANN*), D. Colis., D. Broadcast (*VLAN*)
 - L. de D. o *Propos.* Espec.
 - D. Atrib. = *Conj.* del *Prod. Cart.*

Ver Esp. Nombre (*Ambito*), *CIDR*

37.1 54. Diseño de interfaces de usuario: Criterios de diseño. Descripción de interfaces. Documentación. Herramientas para la construcción de interfaces.

- **GUI** (Graphical Usr Interf.): es junto a *CLI* una *Interf.* del *Shell*. Los Entornos de Escritorio a veces no permite acceder a todas la *Config.* del SO. *Linux* tiene el X11 (X Window System ver *SSH*). Ej: GNOME GNU o KDE.
 - *Dis. UI vs UX (Usr Interf. vs eXperience)*: el UI se centra más en la superficie y el aspecto general de un diseño. Ver *HTML*.
 - *Req.*
 - *NUI (Natural Usr Interf)*: Evol. de GUI (Voz, gestos...)

Ver Mod. *Vista-Control., Event., POO, Smalltalk*. *QBE, Java. Visual. App Inventor, P5.JS (JavaScript)*

- **Vista** : parte de la *Arq.* en *Ing.* del Softw.
 - *Web*: Mod. V. *Control*: *GUI* para *Apl. Web*.
 - *BD*: es *Unif.* por permitir *Compo.* *ner Consult..* Ver *Op. BD, Arq. ANSI/SPARC*

Ver L. *Visual, GUI*

- **App Inventor** : *IDE Apl. Web* para crear App con *Scratch* para *Móvil* (A. o iOS). Basado en *Parad. Progr. Event..* Ver *Android, RT*.

37.2 56. Análisis y diseño orientado a objetos .

- Ver T27POO y *Dis.*

37.3 57. Calidad del «software». Factores y métricas. Estrategias de prueba.

37.3.1 Control de versiones y calidad

- **Calidad** : parte del *Desarr.*, conseguir *Robust.*, buscar *Vulner.*, *Err.* Ej: en *Videojuego* (atraviesa paredes, estresante por deadlines)... Ver *Test Caja Blanca, Certif., Vers., Manej..*
- Contr. **Vers.** (CVS): basado en edición *Concurr.ente (Fork)* y mezcla y *Journal*. En *Despl.* tenemos varias etapas de *Public.* (parece *Tick-Tock* de *Intel*):
 - Alfa: *Test Calidad* por gente distinta a *Progr.amadores* pero del Dpto, en la que se agregan las nuevas características
 - Beta: Test por *Client.* (Beta-Tester), se eliminan errores activamente.
 - *Estable*: donde se han quitado todos los errores importantes

Ver *Dir.*, *Rut.*, *Crit.*, *Git*, *Depend.*, *Unicode*, *Agil.*, *Cript.* Clave P. *Public.*

- **Public.** (Release, Liberar (*Libre*)): un Progr. es parte del *Despl.*. Se pueden P. *Vers.* Alfa, Beta,... Hoy se *Distrib.* por *Internet* como *GitHub*. Ver *Cient.*, *Proyec.*, *Docum.*, *Priv.*
- **Git** : para Contr. *Vers.* *Distrib.*. Permite Dis. *Cooper.* centrado en *Usr* manteniendo las *Coher.*. Otros DP.: Reddit-picture, *Dis.* Top-Down, DP asistido por Computador. De Linus Tolvard y *Libre Hist. I.*
 - **Master** Branch o Rama Maestro: creada por defecto con *Comando git init*. Ver *Fich. Persist.* vs *Tmp* (*Acces. Secuenc. SAM*), *Intel*, *SO Multiproc. Config.*, *Arq. P2P* o *Client. Servid.*
 - **Commit** vs Push: commit guarda los cambios hechos en el repositorio local pero no en el *Remot.*. Push hace la *Sincr.* remota. Ver *SGBD*, *Transac.*, *Journal*

37.3.2 Factores y métricas.

- **Metr. Softw.** : es un tipo de *Metr.*
 - **Eleg.** (no confundir con *Efic.* u *Opt.*): *Metr. Softw.* de código no *Redund.*, que es más *Compr./legible*. Mejora el Ciclo *Desarr.* (*Manten.*, *Depur.*, *Opt....*). Gracias a: *Dis.* Top-Down. Ej: **Cod. Espag.** que ocurre por el *GoTo*. Es un *Unif.* Ver *Repres.* Canon. Rel. *Equiv.*, *TAI*, *Python*, *Estr.. Probl.* Yo-yo (*Her.*), *Monit.* *Semaforo*.
 - **Efic.**
 - **Portab.** : independ. del *Hardw.* donde se ejecuta. Normalmente escrito en *C*. También: *Java* con su *Maq. Virt. JVM* para que sus .jar corren en cualquier SO. Ver *Bibl.*, *Applet Java* o *ActiveX* (*Progr. Compo.*), *POSIX*
 - **Escal.** : que puede crecer. Ej: Motor Busqueda (*Google*) que crece en *Usr* y *Webs*. Ej: Tabla *Rut.* es E. si crece como Mundo Pequeño *Seis Grados*.
 - * E. Vertical (si al añadir *Recursos*, S/T (MS/CPU) a un solo *Nodo* mejora *Vel. Acces. Masiv.*, fácil en *SQL*).
 - * E. Horizontal (si al añadir más *Nodo* mejora en conj., fácil en *NoSQL*).

Ver *BD Nube*, *Kubernetes*, *Manten.*, *Biolog.*, *Evol.*, *XFS Sist. Arch.*

- *Estand.*: código estándar facilita la *Portab.* y *Reus.* Cod.
- Otras: *Reus.* Cod., *Acopl.* y *Cohes.*
- Otros: *Popular*, Curva *Aprend.*

37.3.3 Estrategias de prueba.

- **Test** :
 - T. Caja Negra o Batería de Pruebas: solo ciertos casos. Es Evidenc. no *Dem.* (*Cient.*): Tabl. *I/O* (ej. raices Ec. 2º Grado), *Autom. Ejec.* (con *Script*) y ver que da lo esperado.
 - T. Caja Blanca: con *Lean* o *ATP* o *Iter. While*, para garantizar Sist. *Segur.* o *RT* (avión, airbag), alta *Calidad*

- Otros T70 T. Red (*Cuello Botella*):

Ver *Benchmark*, T-Markov-Mills (*Desarr.*), *Depur.*, *Herram.*, *Aprend.* *Autom.*, *Anal.*, *Monit.* (*Auditorias*), *Certif.*, *Instr.* TSL (*Test Set Concurr.*), T. *Plan.* (*U RT*).

37.4 59. Gestión y control de proyectos Informáticos. Estimación de recursos. Planificación temporal y organizativa. Seguimiento.

37.4.1 Gestión y control de proyectos Informáticos. Estimación de recursos.

- *Proyec.* Progr.: *Docum.* que explica sin ambigüedad como *Impl. Fisic.* una idea. Consta de 4 apartados:
 - *Obj.* y justificación: tras un *Anal. Creat.*
 - Materiales y *Precio*: presupuesto.
 - *Plan.os* y explicación para *Impl. Instal.*: *Graf.*, *Diagr.*, *Topol.* Red, *Esquema*, *Descr.*
 - *Secuenc.iación o Temporalización*: *Rut.* *Crit.* (con *Diagr.* Gantt ver abajo). *Plan.ificación* según *Recursos* humanos disponibles.

La *Iter.* (*Depur.*) en *Ing. Softw.* se hace además de sobre el P. sobre la *Impl.ementación*, en *Ing. Fisic.* solo sobre el P. Otros:

- MS Project (*Win*) o KOrganizer (de KDE Linux): *Herram.* para gestión y *Plan.* de *Proyec.*. También *Backups*, *Simul.* y análisis de *Rut.* *Crit.*. Ver *Softw. Red*
- P. *Softw.*: gestión mediante *IDE* de muchos *Fuentes*.
- P. Colibrí: *Fibra*
- P. Matem.: *Alg.* *Rel.* P.=Idempot *Axiom.* *Boole*
- P. Cooper: si *Modul.-Concurr.* (ver *Caos Aleat.*)

Ver *Mozilla*, *VA*, *Wittg.*, *Eval.*, *Cuant.*, *Agil.*, *Public.*, P. *BD* (*Alg.* *Rel.*), P. *Geom.* (*3D Render.*)

- Método *Rut. Crit.* (avión, *Rut. Crit.*): dado un *Arbol DiGraf.* de Tareas *Compo.* como en Fig. 1.6 el t. max. en terminar el *Proyec.* es el Camino más largo. Usado en *Concurr.-Cooper.*, Contr. *Vers.*, *Fulkerson*
 - *Diagr.* Gantt: similar a *Plan..*. Otros similares con D. CPM o PERT.
- *Precio* (coste, presupuesto) de un material. *Opt.* según *Recursos*. Ver *GPU*, *Nube* (*Hosting*, *Servid.* *Web*), *Libre*, *Jerarq.* *Mem.* *MS*, *Proyec.*
- Control de *Vers.*: *Git*

37.4.2 Planificación temporal y organizativa. Seguimiento.

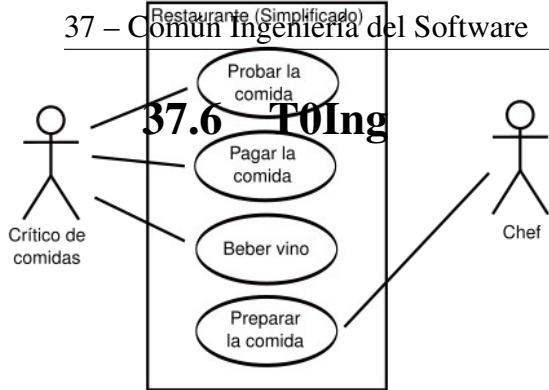
- Softw. *Cooper*, Colab. o CooDis.: es *Concurr.* en *Dis.* (ver Des/Ventajas Des/Ascendente). Gracias a *Modul.* y *Dis. POO (Evol. Abstr. Hist. L.)*. Cuidar *Coher.* o *Consist.* de *Compo..*
 - Acces. *Compart.* a Recurs. con *Grup.* o *Jerarq.* de *Permisos.*
 - *Git*
 - BSCW: Softw. Coolab.
 - Arch. *Compart.*: ver *Ln* (enlace duro *INode*)
 - *Integr.:* se rellenan Esqueletos (*Esquema*) del *Dis.* Top-Down y una vez en semana se reeunen los *Modul.* y ver si se *Compo.* (*Consist.*).
 - Reto: Hackathon (*Agil.*)

Ver *Emerg., Op., Interop., InTranet, Sucursal, Nube, Google Colab (Nube), Evolution (Email calendario), Plan. No Apropiativa.*

37.5 58. Ayudas automatizadas para el desarrollo de «software» (herramientas CASE). Tipos. Estructura. Prestaciones.

- *CASE* (Computer-Aided Software Ing.eniering): Herram. que permite *Dis.* un *Diagr.* UML o *Esquema* o *Cabecera de Cl.* (del Diagr. UML). Pemite *Impl. Autom.* parte de un *BD* o *Progr.* en un *L.* Similar al *Autocompl.*
 - **Umbrella:** Herram. Libre de KDE Linux para pasar de Diagr. UML *Her.* a un *L.* (*Cpp,Java*).

Ver *IDE, Dic. BD, Desarr., Simul.,*



Chapter 38

60. Sistemas basados en el conocimiento. Representación del conocimiento. Componentes y arquitectura. (un poco IA)

38.1 Historia

- *IA*
- Superv. No superv. y Refuerzo (Markov Decision Process)
- Aprend. Refuerzo: basado en *Markov Decis. Process* y usa *Progr. Dinam.*. Agentes Compitiendo *Evol.*: más que un Agente Solo Evolucionando genera creatividad, tácticas inesperadas en Juegos con Aprendizaje por Refuerzo (ver DotCSV)
- Hinton: AlexNET Alex Krizhevsky (Ucraniano), Transformer?? (Toronto y Google)
- Bengio: GANs
- LeCun: CNN
- RNN → LSTM (Hochreiter, Jürgen Schmidhuber) → Transformer (Hinton??) ver *Recurs.* para NPL

38.1.1 Inteligencia Artificial y Bioinformática

- *Dat. Masiv.* (Big Data, o Gigante o Enorme): mayor que *MP* o *MS*. No solo *Almac.* si no extraer *Conoc.* o *Aprend. Autom.* para *IA*
 - *Mem. Virt., Acces. M. (SGBD)*
 - IA con *BD* o *Mem.: Traduc.* basado en *Busq.* I: palabra+ctx y O: palabra-trad.
 - *Almac. M.*: como *Preserv.* en *Mem..* Ver *NoSQL*
 - *Visual Dat. M.*: una imagen más que mil palabras, *Est.*, Ej: Tablau o Mondrian.
 - *Busq.* o *Acces. M.*: ver *SAM...*

Ver COBOL, NoSQL, Supercomput., Apache Spark, Pot., DMA. Fich., Escal., Alta Disponib., Comp. Distrib..

- **VA** (Visión Artificial): *Robot*
 - Homogr.: *Probl. Inv. de Proyec.*
 - Stable Diffusion: revoluciona *Libre2.0* dejando sus modelos entrenados con 256 GPUs Nvidia A100 en Amazon Web Services en 150.000 horas de GPU, con un coste de 600.000 dólares *Hist. BD*

Ver Videojuego, RT, Fire

- **NLP** (Natural Language Processing):

- Chat-GPT: tiene 3 fases de Entren.: Preentren (SGD), AjusteFinoSuperv y RLHF (Reinforcement (*Refuerz.*) learning from human feedback, que es lo que ha sido su breakthroug: [El TRUCO que dió inteligencia a ChatGPT])
- WordEmbedding: One *Hot Vect.* (0s salvo 1) Vect Rey + OHV Mujer = Reina (Word2Vec) (dist. *Semant.*, *Cluster*). Ver *Cod.* (*Combin.*)
- *Wolfram L.*
- SHRDLU LISP: *Sintax. Log.* 1 como DIRHA $\forall x, Apagar(x)$. palabras como bloque, cono, y pirámide, verbos como pon, mueve y deja y adjetivos como pequeño, grande y rojo

- **Recon.** o **Discrim.** Autom.: *Pars.*, Mod. *Est.*, VAD (Voz Activity Detection), GUI NUI

- Bottom-Up (*Dis.*) o SFD: la *Compo.* de distintas *Capas de Sintax.* (*Autom.*) y la Irreduc. *Wolfram* (correr para ver combinaciones) hacen que solo tenga sol. *Progr. Dinam.* (creo tiene que ver con *Lin.* en *Concurr.*). Como *Pila Protoc.* de *Sens.* (señal: oido-oido, espectro: coclea-coclea, *Sintax.*: cerebro-cerebro)

Ver Kinect (Videojuego). Err. Checksum, Voz, AntiMalw., Intr. IPS

- **IA** : Ver Aprend. Autom., Topos. En Centro Dat. poco Ecolog.

- Agente Inteligente: pertenece a la Tribu *Log.* de Domingos.
- **Aprend. Autom.** : según [Domingos] $I = (IData, ODat) \rightarrow Comput. \rightarrow O = (Alg., Conoc., Mod. Prob. o Mod. Est.)$ Fig. 1.6 (como Mem. Asoc.). Las 2 etapas son **Entrenamiento** y **Test.**
 - Aprend. por *Refuerz.*: ver Chat-GPT (*NLP*) y *Markov Decision Process*.
 - *Gener.* o *Prod.* (*G. Proced.* en Videojuegos) vs *Discrim.* (*Recon.*).
 - Entrenamiento: es *Filos..* de *Opt.* Predecir Dat. *Perd.* (ver). Basado en *SGD*. El *Conj.* de Entren. tiene *Dat.+Etiq.*
 - *Etiq.* (label): menos estricto que *Jerarq.* (*Grup.*). Ver *HTML*, *XML*, *CSS*, *Sist. Arch.*, *Var.*, *Id.*, *Direc.*, *MPLS*, *Patch*, *Docum.*, Clave (*Key*) *Hash*, *Ind.*, *Arbol*, *Dic.*
 - **SGD** (Descenso en Gradiente Estocásitico): Met. *Cient.* *Iter.* de Newton (derivada actual) *Est.* para *Opt.* parámetros F. perdida Multidim (entrenar *NN*)
 - **GAN**: se puede ver como un AutoEnCod.er o *Compr..* Parecido a *Cuantiz.* *Vect.* (*VQ*). Ver No Repudio (*Autent.*)

- **PGM** (Red Bayesiana,...)
- Otros: *Vect.* SVQ, DTW (*Progr. Dinam.*), *PCA*
- *Bibl. Python*: Pytorch and TensorFlow. Numpy. Machine Learning for Kids, LearningML
- *Autocompl.* : GitHub Copilot y Alpha Code. La Program. *Autom.* es imposible según *Goedel* pero si la *Traduc.* (decir en L. Natural y pasar a Codig.). Ver *IDE*, *CASE*, *ATP*.
- *Evol.*
- *Err.*: debe lidiar con el Ej: *QoS* malo
- *Cluster*: aGrup.ar *Geom.* los datos. Buscar Clique *Graf.*. Es *Acopl.* y *Cohes.* forma de *Conoc.*

Ver Curva *Aprend.*, *Exten.* AMX, *Model.*

- *Transf.* (Transformer): se base en atención y T. Fourier 00000000 (11111111), 00001111, 00110011, 0101010101. Paper de *Google*. Ver *Graf.*
- *Bosque* s *Aleat.* (Random Forest): *Aprend.* *Autom.* basado en crer mucho *Arboles* de Decisión.
- *SPN* y *NN* : son ejemplos de *Concurr.*, *Paral.* y *Ord.* Parc.
 - RNN: ver *Recurs.*
 - NN: Maq. Boltzmann (*Topol.* Malla) es como Hopfield (*Asoc.*) pero usa *Muestr.* Gibbs annealed en lugar de *SGD*
 - Def. SPN: como *Obj. Recurs.* (Obj. que toman como I. otros Obj.) o *Dis.* Bottom-Up empezando por Hojas *Arbol.* Fig. 1.6. Sc: scope o *Var.* a modelar. $[0.20.8] = [\mu\sigma]$

Ver *SGD*, Curva *Aprend.*, *Diagr.* UML (*Her.*)

- Empresas de IA: **OpenAI** (de Elon Musk, GPT3, DALL-E (usa GPT3), Chat GPT-3 GitHub Copilot) (ver *NLP*) **DeepMind** (de *Google*, LaMDA (Lemoine), AlphGo/Fold-/Code (*Autocompl.*), DeepDream, TensorFlow).
- *Robot* : *Comput.* de Sist. *Fisic.*
 - *TeleRobot*: deberia enviar la información con estandar de video juegos del momento (ahora Realidad Virtual, ver *JSON*).
 - *Sens.* (tiene un A. *Banda*, ver *Analog.*) y *Actuad.* . *Transduc.* (conversor de un *Tip.* *Energ.* a otro pero de *Signal* mucho más pequeña) ver *Mem.*)=*Transcep.*=*Traduc.* Ej: S. Dist. Ultrasonidos HC-SR04 (T/R, parece ojos). Son *Unif.*. Ver *Dat.* *IoT*
 - *Driver* de Motor (Electr. de *Pot.*): Puente H de motores DC L293D (negro y pequeño) y L298N (rojo y grande)
 - *Conect.*: Jumpers, Pinzas cocodrilo. Interesan que sean *Hot*.
 - IDEA: usar materiales con *Mem.* como el NiTiNOL para *ALU Recurs.* *Biolog.*

Ver *Autom.*, *Perif.*, *Dispos.*, *Brazo-Disco*, *Arch.* *Dispos.* (*Socket* o *Serie*), *VA*, *IoT*, *RT*

- *Bioinform.* : Alineam. Prot. *Progr. Dinam.*, Ver *Turing*, Chaitin (*TAI*)

Chapter 39

Común Redes

- Comput. *Nube* (Cloud Computing): para cualquier *TAP* a través de un *Servid.* contratado por un *Precio* o gratis. Un *Admin.* *Virt.*liza y *Encapsul.* (*Transpar.*) los *Recursos* que ve el *Usr* restringiendo *Permiso*. Dichos Recursos pueden estar *Distrib.* en varios Servid. o *Centro Dat..* Para *Almac.:*
 - *Protoc.* NAS o SAN (para *LAN*)
 - Ej: Dropbox, *Google* Drive, iCloud, *Amazon AWS*, *Win Azure/One-Drive*

Para *Proc.:*

- AWS (*Amazon Web Services*): para Entrenar Stable Difusion VA).
- Microsoft (*Win*) Azure (ver *Ubuntu Server SO Servid.*)
- *Google Cloud* o G. Colab (*Cooper.*): para ejecutar *Python GPU* y *Aprend. Autom.* en la *Nube*:
- *Hosting (Servid. Web)*
- *Almac. Distrib.* o *BD Nube.*

Ver *Kubernetes*, *Jerarq. Mem.*, *Grid*, *SO N.*, *Sucursal*, *BD Nube*, *Almac. N. Jerarq. Mem.*, *Backup (Preserv.)*, *Almac. Distrib.*

- *Tele* (presencia/transporte): 1a ventaja de *Red=Transpar.* (*Topol. Red*). Nod. *Acces.* independ. de su localización *Geom.* de forma *Transpar..*
 - **Telegrafía:** Nyquist A. *Banda*, *Concurr.* ferrocarriles *Hist. SO*. *IoT=TeleRobot*. *TeleMusic.* (*RT*) Telescopio (LIGO *Interfer.* *RT*). Teletrabajo (ver *Remot.*). *Telemedicina* (*RT*)
 - *Telef.* : su *Red* con Cobre transporta *Voz* y datos (*Modem*). Usa 1 *Par Trenz.??* (*ADSL*) con *RJ11* aunque en Alemania *RJ45*. Puede ser *Conmut.* *Circuit* (*RTC*) o *Básica* (*RTB*). Orden de aparición de *Servic.:* 1º Voz en *RT*, 2º llamadas en espera, conferencias a 3. 3º *Dat. Internet* (por *DSL* y *Servic. Int.*) Ver *Multiplex. Tmp. PPP*.
 - *Movil* : pequeño *Comput.* como PDA (Personal Digital Assistant), Smartphone, o *Tablet*.
 - Arq.: tienen *RISC*. *Display* flexible, *Mem. Tarj.* Micro SD (*Flash*, no confundir con *Tarj. SIM* para la Red ni *SIMM (RAM)*).
 - Dat.: CELP *LPC*

- SO: *Android*
- L.: *App Inventor Event.*
- Red: torres *Antena* elevadas. *WWAN* que usa los *Protoc.* de comunicación de abajo, *Colis.. Roaming* (itinerancia, ver GSMA Open Gateway *TIntegr.*)

Protoc.:

- 2G **GSM** (Global System for Movil): transmite *Voz* a 9,6 kbps, de la *ETSI* con *CRC*, puede *Suplant.* Usa *Commut.* Circ. *Multiplex.* TDMA
- 3G UMTS: extensión 3G de GSM
- 4G LTE: 4G, usa *Modulac.* OFDM con *Transf.* FFT. *RF* a 2-8GHz (como el *Wifi*, SHF Super High Frec. entre UHF y *Microondas* bajas Fig. 1.6) A. *Banda* de 10 MHz. Creo es *Red Bus*.
- 5G: sin estandar. retrasado por COVID. 4G y 5G son *C. Enl.??*
- 6G: para 2028
- Establecen: *RF* y *Microondas*, Mbps. en *Antena* elevadas (Fig. 1.6). *Colis.* por *Red Bus*.

Ver *Esper.* Activa, GSMA Open Gateway (*Integr.*)

- *TV: Digit.* Terrestre (TDT), *Stream*, *Triple Play*.
- **WiMAX** (IEEE 802.16): *WMAN (Wireless)* que usa *Microondas* hasta 70Km. En zonas rurales donde no existe terrestre. Relacionado con *Wifi*
- **Satelite** : hace de *Nodo* en *WAN* privadas o publicas. StarLink de Elon Musk o VSAT. Hay *Microondas* GEO/MEO/LEO. Ver *TV DVB-S (Broadcast)*,
 - GPS: de DARPA83 *Hist. R., RT*
- **Wireless** : *Medio Transm.* inalambrico (no guiado) que usa *EM (RF, Antena..)*. Gracias a las *MOSFET RF* F. Tab. de *Ext. Red* Inalambr cada nivel casi su IEEE 802. Más *inSegur.* e *Intercep.* que *Cableada* (*WPA, Medio* abierto)

WPAN	<i>Bluetooth, ZigBee, Infrarrojo, RFID</i>
WLAN	<i>Wifi WAP, Lifi</i>
WMAN	<i>WiMAX</i>
WWAN	<i>Movil (1G-5G)</i>

Ver *Lifi*.

- GSMA Open Gateway (*Movil*): *API Libre* de *Linux Foundation* y *Proveedores* principales (Orange, Telefonica,...) para abrir las redes a los *Desarr:adores* y conseguir la *Web3.0*. Para *Integr:ar* Servic. en la *Nube* a todos los *Moviles*. del mundo, *Transpar.* al Operador **similar al Roaming** (que si hizo en 87 para *Integr.* la *Voz Hist. R.*)
Las 4 Op. *Movil Europ.*: Telef., Vodafone, Orange y DeutscheTelekom critican las grandes

*Empresas Tecn. (Google,...) se aprovechan de su infraestructura y no pagan lo suficiente [El País]*¹

- JUANBE INTROS:

- Tirar de lo SOCIAL, antes de Tinder se ligaba, nunca se planeó esto. Eran ingenieros frios que no lo esperaban. En telegrafo. Ley de Moore similar. Telegrafo, trenes igual. Antes un telegrafo mensaje=sueldo un profe mensual. Modelo OSI, surge como modular código, en red para no cambiar todo el hardware, si Conector físico con 2 o 3 no tengo que cambiar el resto. OSI nace para poder reusar. Cada vez que haces un SWITCH no cambiar todo. El código entero de OSI 1MB, pero si partes y en Switch solo implementas los protocolos de 2 primeras capas no ocupa tanto. En 1 Linea: vamos a ver esto....
 - En 2 hojas entran todos los temas de redes. Apuntar unas cuantos protocolos y decirlos entre medias del texto, pero no tablas y listas tochas. Y hacer los que creas: ej. Sist. Informáticos = Sist. en red y enrollarte.
- **HTML** : L. basado en SGML al igual que **XML**. UI vs UX (ver *GUI*). Todas las *Etiq.* tienen inicio y cierre *<h1> </h1>*. Es *Imag. Vect.* Ver *HTTP*
 - Canvas: ver *JavaScript*.
 - *Web Semant.*: gracias a *Etiq.* como header o article que no tienen impacto *Visual* pero si en *Naveg..*
 - HTML5: permite *Multimedia* y *Web2.0* como P5.JS *JavaScript*. *LiveStream*
- **CSS** (Cascading Style Sheets): donde definimos el *Form.* para todas las *Etiq.* *HTML*. Evita poner *style* en *HTML* haciéndolo en otro *Modul..* Se *Enl.* al *HTML* con *link*. Ej: *h1* (head1) en rojo y centrada siempre. Fondo background-color rgb. Hay una *Jerarq.* en caso de ambigüedad de *Ambito* (ver). Permite *Sobrecarg.* de *Etiq..* Ver Cascada=Busq. *Binar.* (ISAM)
- **JavaScript** : por [Eich95] en *Empresa Netscape*, ahora marca de *Oracle Hist. R..* El 95% de *Frontend*, reduciendo demanda a *Servid.* aunque es quien le pide datos. Es *JIT (Interpr.)* y *Parad. Progr. Event. (mouseX/Y)*.
 - **AJAX** (JavaScript ASincr. y XML, Asynchronous JavaScript And XML): apl. ejecutada en *Client.* mientras mantiene comunic. ASincr. con *Servid.* en 2º plano sin necesidad de recargar la *Web*. Ej: *Google Maps*. Envía los datos *JSON* o *XML* *Seriellizados* (permitiendo *Persist.* [Serie]). Trabaja con *Bibl.* *jQuery (Consult.)* y *P5.JS*
 - *Progr. Fun.:* F. Anónima para *Curryficar* (ver ej.)
 - **JSON**
 - **P5.JS:** *Bibl. Popular* similar a *Applet Java Processing* para *GUI, Imag. Vect., Graf. 3D*, o *Videojuego*. Trabaja con *AJAX*.

¹se lanza con ocho redes universales de API, incluyendo intercambio de SIM, QoD, estatus de dispositivos, verificación de números, facturación o localización, entre otros, Permite localización, emisiones en vivo a través de 5G; o juegos inmersivos y vídeos de alta definición de la mano de Telefónica, <https://cincodias.elpais.com/companias/2023-02-27/lasgrandes-telecos-mundiales-se-alian-para-abrir-las-redes-a-los-desarrolladores.html>

- * **Canvas**: las *Applet Java* se ejecutaban en *Naveg.* usando el *JVM Hist. R. Hist. L.* (ver ActiveX Progr. Compo.). Los *Naveg.* dejaron de soportar su Ejec. en 2015 (con Java9). Con la llegada de **Canvas** en *HTML5* se permitió a *JavaScript Render. 3D* usando *GPU* (mediante Bibl. 3D Open/WebGL) y ser tan *Vel.* como Java. Ej: *import processing.core.PApplet;* de Processing (*JavaScript*) recuerda esa época. *RT* sin atranques sería ideal
- *Bibl.* o Frameworks Populares JS: React (no confundir con ReactiveX Event., permite Progr. Compo.), Angular.JS, Node.js (ver), P5.JS (ver), WebGL (Canvas Render. 3D), jQuery (AJAX), y View??
- **Node.js**: framework para programar *Backend* con JS, compite con *PHP*. **React.js** es una *Bibl.* de JS para crear *Interf.* de *Usr* en el *Frontend*.
- *Empresa*: según encuestas stackoverflow sueldo anual promedio es 60000 dolares.

Ver *JSON, 3D, WebUSB*

- **PHP** : Input: URL *Web*, Output: *HTML*. Para *Backend*, compite con *Node.Javascript* y *Python*.
 - Pas. *GET*: se envian Inputs por URL con Key+Valor (max. 2000 caract.)
 - Pas. *POST*: se envian Inputs ocultamente, permite enviar cualquier Dat. *Binar.* o *Multimedia (Audio, Imag.....)*
 - Comando *MySQLi (Persist.)*.

Ver *PhpMyAdmin (MySQL), SQL*.

- **Naveg.** (Explorador, Browser) *Web*: Para *Busq. Info*
 - **PageRank**: *Busq. Masiv.* de Larry Page *Google Hist. R.*, adoptado hoy por resto de *Naveg.* Relacionado con Search Engine *Index* (Factor *Block*)
 - N. *Web Popular*: Chrome (*Google*), Safari (*Mac*), Edge (*Win*) y Firefox (*Mozilla*).
 - **Mozilla** : nace por la *Guerra N. Hist. R.*. En 1998 viendo que N. de *Empresa Netscape* perdía frente a *Explorer*, este se hizo *Libre* su código originando el *Proyec. M.. Ver Thunderbird Email, SuperCoockie*

Es *Unif.* por *Topos (Busq. Binar.)*. Ver *Admin. Arch.*, *MD5 (Cript.)*, *Applet, Coockie, BD N. (BD Jerarq.)*

- **Cookie HTTP**: trozo de información puesto por un *Servid. Web* en Comput. *Client.* (en *Naveg.* de un *Usr*) para futuros accesos (como guardar *Passwd* o hacer *Spyware Malw.*).
 - SuperC.: *Mozilla* lo llama a la *Cache* del *Naveg.*
 - *HiJacking* (secuestro) *Sesion*: se roba la C. Magica utilizada para *Autent.* a un usuario ante un *Servid..* Otros *Hijacking* son: *Defacement* (desfigurar *Web*). Ver *Malw. Ransomware*.
- **Apl. Web** : Apl. que en lugar de correr en *SO* lo hace parte en *Navegador* (Cliente: *JavaScript*) y parte en *Servid. Web*
 - *Form.* (Formulario): con *Campos*, relleno *Autom.* con *Scraping*.

Ver *Vista*

- **Servid. Web**: Servid. de Webs que acepta Petic. HTTP y Apl. Web. Usan Apache o Win MS IIS. Ver *Cookie, Daemon*.
 - **Apache**: el Servid. Web HTTP mas Popular. Otro es Nginx (Email) o MS-IIS (Stack Soluc.) o XAMPP Tiene 30 Hilos de la aplicación PHP esperando a un Client., si se conecta más, se crean más. Ver A. Cassandra BD Distrib., A. Tomcat (Arq. MultiTier BD), A. Spark BD Distrib.
 - *Proxy*:
 - *Stack Soluc., Mashup, Scraping*
 - *Hosting*: Servic. de Alojam. Precio gratis o por més de pago para tener Comput. Web. Servid. Web con Apache y Stack Soluc. (con Blog WordPress..). Los hay gratis como 000webhost o web2drive, y locales como XAMPP. También Comput. en Nube Amazón, Goolge Colab. Ver Cuota Disco (Sist. Arch.).

Ver FreeBDS, Applet

- **Applet**: programa que puede Embeb. en un documento HTML (página Web). Cuando un Naveg. carga una Web que contiene un Applet, éste se descarga en el navegador y comienza a Ejec., lo cuál nos permite crear programas que cualquier usuario puede ejecutar. Cuidar que no sea Malw.
 - **Servid. Apl.** (SA): Servid. de Apl. que corren en Backend en lugar del Frontend (Client.). Cuidar el Equil. Carg.. Ej: **Servlet**: Applet definida por Jakarta EE (Progr. Compo.) que corre en el Backend. Es una API-Web para para Proc. Petic. del Frontend y enviarle un HTML (Actual. de la Web).
Otros Ej: Wolfram o de Videojuegos (ver Minecraft).
 - A. Java: en desuso (ver Canvas JavaScript) y ActiveX (Progr. Compo.), Hist. L.

Ver Servid. Web

- **Blog**: el mas famoso es WordPress. Ver Ataq.
- **IoT** (Internet de las Cosas): es Tele-Robot o Compo. de Recursos.
Red de Transduc./Perif. TeleCompart.. debido a la diversidad de Dispos. se puede Impl. como una Red Superp..

- Protoc.: BLE (Bluetooth Low Energ.), ZigBee.
- Hay Win y Ubuntu IoT.
- **ZigBee**: IEEE 802.15.4 para WPAN (Wireless) de IoT Domótica
- WebUSB: JavaScript API para IoT de Perif.
- Google-Home: encender Luz led con Wifi
- MiraCast: Wifi Direct
- ThingSpeak: le envias Insert. por URL Web.
- CiberSegur.: coche, RT (con RTP), TeleMedicina
- Comput. Frontera (Edge C.): tipo de C. Distrib. o Topol. Red donde se acercan los Recursos y Servid. al Client. para mejorar T. Resp. y ahorrar Ancho Banda. IoT es un tipo (pero no el Recipr.).

Ver *Embeb.*, *Cable Estr.* (*Par Trenz.*), *WakeOnLAN*, *P2P*

- **Transcep.** =*Transduc.*: *Dispos.* que puede *Transm.* y *Recibir* (*semiDuplex* pues no puede hacer las 2 a la vez). Realiza conversiones entre tipos de *Signal*. Ej:

- *RF* de aficionado: *Walkie-talkie*, *Antena...*
- *Repet.*
- **Balun** (*Balancing Ud.*): *Conect.* que pasa de *Coaxial* a *Par Trenz.* sin afectar las *Imped.*)

Es un *Unif..*

39.1 73. Evaluación y mejora de prestaciones, en un sistema en red. Técnicas y procedimientos de medidas.

- *Centro Dat.*, *Patch*, *Rack*
- *Disponib.*

39.2 74. Sistemas multimedia

- *Hardw.*: *Tarj.* audio, *GPU* Todo *T Perif.*
- **Multimedia** : centrarse en:
 - a) *RePresent.. M.*
 - b) Sist. *Transm.* y *Stream M.* más que en la *) ESTE NO *Progr. M.* (*P5.JS JavaScript*) o *Cod.-*
 - los *Fich. M.* llevan una *Cabecera*.
 - *Cod.ec*: programa o hardw capaz de de/cod. (des/*Compr.*) un flujo de datos (*Stream*) *Multimedia*. Ej: H.264 =AVC (ver *MP4*) y G.711 (*Cuant. Voz*)
 - **Container** : es *Form.* para *Embeb.* en 1 *Fich.* varios *Fich.* de diversos *Form..* Tiene una *Cabecera de Metadatos*. wrapper o metafile Fig. 13.11 Ej: *ZIP*, *AVI*, *MP4*, *PDF* . Otro significado ver *Virt. SO*, *DUD*.

Ver *Stream M.* (*RTP*). *Compr.*, *Web*, *Videojuego*, *HDMI*

- **TV** : *Broadcast* o *Streaming* de *Video*:

- *TV Digit.* Terrestre (TDT) (no confundir con *TV* por *Cable*): *transm.* no *Analog.* pero en *RF* (con *Antena*) y *Broadcast*. Empezó en 2006 *Hist. R.* y en España sustituyó a la antigua en Abril de 2010, ya está en casi todos lados. En *Europ.* y África tenemos DVB-T (*Digit. Video Broadcast-Terrestrial*) que usa *Modulac. COFDM* y *MPEG2*. DVB-S es con *Satelite*. *TV* de antes de 2014 necesitan *DeCod.*, si no basta la *Antena Recep.* conectada directamente, esta puede ser interna (pequeña y al lado de *TV*) y externa (en tejado) ambas en *Direc.* a *Antena Emisora*. Luego hay que *Sintonizar* la *TV*. Es un poco más interactiva como la *Web2.0*

- DAB (*Digit. Audio Broadcasting*): es equiv. a TDT TV Digit. Terrestre, pero no tiene éxito y seguimos con *RF FM*.
- Smart TV (no confundir con TV Digital Terrestre): TV con *Internet* y *Web2.0*. Es Converg. *Tecn..* Va a reemplazar a Noticieros de TV Digital. HbbT (*Hybrid Broadcast Broadband*): mezcla de SmartTV+TDT (ej. mensajes de pincha para TV a la carta de CanalSur y TVE).
- TV por *Cable*: *Cable* directo a TV *Coaxial* con *RF* o *Actual. Fibra* con *Puls.* de luz (sin Antena, sin Broadcast). Dicho *Cable* suele permitir *Triple Play*.

Ver *MiraCast. RTP*

- ***Triple Play*** : TV por *Cable*, datos *Internet* y *Voz Telef.. HFC* el 1º en permitirlo, similar al *DSL* y *Servic. Int..* Es Converg. *Tecn.* [Network convergence]. Ver *Streaming*
- ***Stream*** :
 - RTMP (*RT Messaging Protoc*): de Adobe Flash Player
 - Non-Live Stream: Ej: Youtube, Video Bajo Demanda o *Web* con video. Se hacen con *HTML5* ver ej. abajo.
 - LiveStream (no confundir con): es S. en near-*RT*. Ej: *Robot* en Marte o Twitch
 - Basados en *Broadcast, RTP/UDP* y *UniCast* o *IGMP (IGP)*.
 - *Video Bajo Demanda (VBD, TVoD)*: Netflix, PrimeVideo.

Ver *Triple Play, MPLS. MiraCast. Triple Play, Multimedia, VoIP, Caden..*

```
<video src="movie.webm" poster="movie.jpg" controls>
    This is fallback content to display for user agents that do not support the video tag.
</video>
```

- ***VoIP*** : Transm. *Stream* de *Voz*, SMS o fax en *Internet (UDP)* en lugar de red clásica *Conmut. Circ. del Telef..* Creado por Sincoskie (*VLAN*). Para garantizar un *QoS* y evitar Probl. del Mejor Esfuerzo [*VoIP*], se basa en *MPLS, UDP* (usualmente). Ver *RTP, Stream, Ethernet*
- ***Sindic.*** : redifundir un contenido de otro propietario, bien por haber pagado los derechos o bien por Licencia *Libre. Web* (*Blog*) que resume/muestra automáticamente noticias o contenido que cambia en otra Web. Ver *Mashup*
- Motor ***Videojuego*** s: Gráfico Espacio Temporal Interactivo usando *GPU (Video+3D)*. Unreal Engine (*IDE Visual Studio*)). Los *IDE* están especializados para trabajar con un Motor de V. específico.
 - ***WYSIWYG*** (What You See Is What You Get): *HTML, Latex, GUI*
 - *Hist. L.*: Pong: su éxito en los 70 despegó la industria de los V. Se hace bien con *POO*. “El Ajedrecista” *Torres*. Fogar 1981 de Sega (*POO*).
 - FIFA o Fortnite: se hacen con *Cpp*.
 - *Gener. Proced.*: de Dat. como Texturas *Fract., Vect.* en Minecraft. Futuro *Audio* con *Music.LM [DotCSV]*.
 - Sprite (duende): *Obj.* (muñeco como Mario Bros. o Prince of Persia) de V.
 - Videoconsolas *Populares*: Xbox (*Empresa Win*), Nintendo y PlayStation (Sony) ver Zocalo (*CPU*)

- Kinect/Wii (*Sens./Recon.* de Voz, Mov.para Xbox/Nintendo)
- L. V. 2D: con Sprites (fotos de personajes): *Python Game*, *App Inventor*, Godot (Gartzen)
- L. Graf Vect.: Python Turtles, P5.JS *JavaScript*.
- *Servid. V.*: Minecraft. para compartir mismo mundo. a) Servidor: Instalar *JVM* y lanzar SpigotServer1.16.jar., abrir puerto 25565. b) Clientes: instalar *JVM*, lanzar LauncherFenixClient1.16.jar (misma *Vers.*) y poner IP-Server.
- *Bibl. C*: BGI u OpenGL (ver *Render.*).

Ver *Multimedia*, VA, *Calidad*

- **Render.** (Síntesis de *Imag.*, no confundir con *Rend.*): *Proyec.* en 2D un Model. *3D*. Es el proceso de *Gener.ar* una imagen fotorrealista (*Comput. Luz*) a partir de un Model. 2D o 3D mediante un programa. El **Render** es el resultado o la proyección que estamos viendo.
 - OpenGL: *Bibl.* o *API 3D Libre* para Renderizar permitiendo acceso a *GPU*. WebGL de *JavaScript* basado en ella. Es multiplat. (*Portab. Linux/Win*). Otras similares son: WebGL (Canvas *JavaScript*) y **DirectX** (de *Win* usado por ATI Radeon *GPU*, no confundir con ActiveX (*Progr. Compo.*) ni ReactiveX (*Event.*)).

Ver Canvas (*JavaScript*), *Video RAM*.

Chapter 40

*+61. Redes y servicios de comunicaciones. (1OctMarcos)

40.1 Introducción. Conclusión

- Ej-motivador: a) ¿Si la red está muy saturada como seguir ofreciendo un buen servicio?
Sol.: regular *Traf.* según *QoS*
b) ¿Como funciona Skype cuando llama a un Telef.? Sol.: *Triple Play*
c) Qué es *DNS, DHCP*
d) Crec. *Moore*.
- Hist.: ARPANET Red UNO (*PDN*) Telegrafo Wester Union ATM, Frame Relay, DSL GPS DARPA83
- Futuro: Comput. *Nube, Kubernetes (Admin. Servid.)*

40.2 Red

- *Red* de Comput. o TeleComunic.: *Hipergraf.* (*Red Bus*) donde los *Nodos* son *Hardw. Red* (ETD o ECD) y los Arcos son *Medio Transm.* para *Comunic./Compart. Inform./Recursos (Servic. Dir.)*: Ver *Internet, Traf., Biolog., Seis Grados, IoT, R. Acces. (Internet)*
- Bondades de las Redes (características) (*B. [Stallings07]*)
 - *Telepresencia: Acces. Transpar.* (1a ventaja) (*Topol. Red*)
 - *Id.:* de nodos sin ambigüedad. Suele \exists una Id. de *Broadcast* y otra de Red.
 - *Robust.:* *Disponib.* por *Redund.* (único requis. DARPA Net)
 - *Vel.:* A. *Banda y Flujo*

Seis Grados=Libre Escala (el grado de los nodos decae pot.)+Mundo Pequeño (saltos entre nodos crece log. con n° nodos)

40.3 Clasificación de las redes

- Según Propiedad:
 - *Priv.*: con IP cuyos paq. no se escapan a *Internet*. Ej: *IoT*.
 - *Public.* o *PDN*: Op. por *Proveedor Priv.* e infraestructuran *Internet*. Ej. Red UNO *Hist. R.*.
 - Propietaria: ver *PDN*
- Según *Topol.*:
 - Fisic.: *Red Bus* (*Robust.*, *Colis.*), *Star* (*Err.* centro), *Ring* (doble), *Tree* (raro), *Mesh* (mixta)
 - Log.: *Ad-Hoc* (*WAP*), Infraestructura (*Star*), *Mesh* (mezcla, *Home*)
- Según *Ext. Red*: cada *Nivel* casi con su *IEEE 802*
 - *PAN* (Personal Area Network): 2 *Moviles*, *IoT*
 - *LAN*: PCs de vivienda
 - *MAN* (Metropolitanian Area Network): 2 Conserjerias o Paravisa??. Tiene su *IEEE 802*
 - *WAN*: Conecta varias *LAN*. Ej: *Internet*, *Movil*, ARPANET. Construidas por *Proveedor* (WAN pública de pago) o *Empresa* (privada). Ver *VPN*
- Según *Medio*:
 - *Wireless*: WPAN (*Bluetooth*), WLAN (*Wifi*), WMAN (*WiMAX*), WWAN (*Movil 5G*)
 - *Cable*: *Par Trenz.*, *Coaxial*, *Fibra*,
- Según *Comut.* (Switching) (*E2E*): hay 3 formas de *Impl.*ementar una *Red* (de telecom.) Fig. 42.12:
 - C. Mensaje: el mensaje pasa completo de *Nodo* a *Nod.* (sin trocearse). Ej: X.25 (Virt.), Cartas, *Telegrafo Wester Union* (*Hist. R.* 1962 Defecto del Delay *Cola*) y 1er ARPANET. Hoy C. M. se hace con C. Paq.
 - C. Circ.: se establece camino **Dedicado** no disponible hasta fin de conexión. Hay 2 tipos: a) *Multiplex.* en t y b) M. en f. Ej: *Telef.*, *OrCo*, *GSM (Movil)*
 - C. Paq.: partir la info en *Paq..* Es la más usada *Internet* y *Frame Relay*. Hay 2 tipos: a) *Circ. Virt.* (*TCP*) y b) Red Datagram. (*UDP*). Ej: No *OrCo*. Ver *Autom.* Cel. y Sist. *Emerg.*, Red UNO (*PDN Hist. R.*).
- Según Comput.-Apl. *Distrib.*: (ver *TProtoc.*)
 - *Client.-Servid*
 - *MultiTier*
 - *Distrib.*

- Según Proposito:
 -) Red *Telef.*.
 -) R. *Broadcast*. *TV*
 -) R. *Dat.* o *Digit.*.
 -) R. Servicios *Integr.*: todo o parte de lo anterior. *Triple Play* (abajo)
- Otras: a) No/*OrCo* (*TCP*), *AcRe* (*Ack*) y Principio *E2E*
b) *Duplex*, Red *Broadcast*

40.4 Arquitectura por capas y componentes básicos de redes

- *TCP/IP vs OSI: Paq. Encapsul.*
- *Hardw. Red:*
 - *Hub, Medio*
 - *Switch*
 - *Rut.*
 - *Gateway*
- *Softw. Red:*
 - *NOS vs SO Servid.*
 - *Estand. y Protoc.: ITU, ISO, IEEE, IETF*

40.5 Servidor y host

- *Servid.* : en Arq. Client.-S., es un *Host* (conj. comput. o softw.) que ofrece *Servic.* (*TAP* o *Recursos*) a las *Petic.* de otros llamados Client.
Puede usar *Servic.* de otros S. y una misma comput. puede tener varios S..
Hay **No/Dedicado** (ver *SO Servid.*) Tipos:

- *Servid. BD (MySQL)*
- *Servid. Web (Apache)*
- S. Apl. (*Applet Servlet*)
- Otros S.: ver *Servic.*

Ver Arq. Red, RAID (*Redund.*) Controlador Domin. (*Servic. Dir.* Active Dir.), S. Apl. (*Applet*), Truco DNS para tener IP Estat.

- *Host* (o anfitrión): *Comput.* (PC, *Movil*, *Estac.* *Trabajo..*, tabletas) incluso *Servid.* o *Nodo Terminal* (ETD) con una *IP* que ofrece o consume *Servic.*.
Su Ambito de Influencia de Capa es C. Apl.. Ver *DHCP*, *Hosting* (*Servid. Web*).

40.6 Servicios

- **Servic.** (no confundir con F. Capa (abajo) o Capa de SO como *Shell*): F. ofrecidas por Servid. en C. Apl. o más alta en Arq. Client.-Servid. o P2P.

No confundir con F. Capa: A veces entendido como: F. $n - 1$ a capa n de forma *Transpar.*. En Client. accede por una *Interf.* de Usr Id. con un *Puerto*.

DoS: *Ataq. (Inanic.)*

Podemos clasificarlos por *TAP* o *Recursos* (Almac., Perif.) o por *Informat.* (A,SO,..):

- *Servic. Dir.: Impr. y Recurso de Red*
- *DNS, DHCP: Admin. Autom. de Red*
- *FTP, SSHFS*
- *Servid. BD, Servid. Web, Email (SMTP, MDaemon, MSExchange),*
- *Nube y Web2.0: Videojuego, Wolfram, Google Colab.,*
- *Stream: VoIP y Video Bajo Demanda., Radio-web, Teletrabajo/conferenc*
- *IoT: Sens., Perif.*

Otros:

- *Proxy, Escrit. Remot., VPN Sucursales,*
- **SAP** (Service Access Point): localización donde una capa puede pedir a otra capa un Servic. (normalmente capas OSI)

Ver *Servid., Daemon, Cola, BIOS, Middlew., Proveedor*

- **QoS** (Quality Of Servic.): RESUMEN Internet no es RT por Mejor Esfuerzo, luego se Prior. con MPLS para garantizar Metr. Vel. Latencia y Perd.. Esto da las NGN.

- *Metr.* basada en *Vel. Transm., Latencia, Disponib., Perd. Paq., Equil. Carg. (Plan. Proc.)*
- **Prior.** : Control del Traf. según la Apl (E2E). Ej: VoIP, MPLS o resolución automática de YouTube, **NGN**.
Le interesa a *Empresa* que contrata el Servic. Es *Unif.*. Ver *Aprend. Autom., BD, SGBD, T. Resp., PCB, Colas Multin. (Plan.)* y *P. Crit. (Plan. RT (RMS)), P. Dinam. (Plan. RT), Interr.*
- *Probl.* de Entrega de Mejor Esfuerzo [Best-effort delivery]: aquella que no puede garantizar que los datos lleguen a su destino ni un QoS.
Es *Filos.* democrática de *Internet*. y depende del *Traf.* (varia el A. Banda). Esto hace que no termine de implantarse *VoIP*
- **NGN** (Next Generation Networks): es la Converg. *Tecn.* basada en transportar cualquier dato *Triple Play* en red *IP. Unif.* por IP/MPLS. Fig. 40.10.

Es *Unif.* como *RT* (garantizar).

40.7 MPLS

- **MPLS** (MultiProtocol Label Switching, *Conmut.* de *Etiq.* MP): En *Rut.* de un Nod. a otro basandose en una *Etiq.* de *Prior.* más que en la *IP*.
MultiProtoc porque *Integr.* muchas infraestr: puede usarse para transportar paquetes IP, así como *ATM* nativo, *Frame Relay*. Redes (SONET) o Ethernet.
LPS (abajo) porque es para redes pero mientras que las IP identifican puntos finales, las *Etiq.* id. rutas establecidas entre puntos finales.
Trabaja a nivel de *C. Red* y *C. Enl. Capa 2.5*. Es para *Conmut.* *Paq.*
Transgrede (como *VLAN*) Filos. democratica de *Internet* pues garantiza un *QoS* aten- uando problema del Mejor Esfuerzo

- Def. por *IETF* en *RFC 3031*: Su *Cabecera* lleva un contador *TTL* (evita *Circ.*). Fig. 40.10
- LER (Label Edge Router o enrutador frontera de etiquetado)
- LSP (Label Switched Path o intercambio de rutas por etiqueta)
- Esa *Etiq.* permite *Prior.*izar paq., controlar el *Traf.* y garantizar un A. *Banda*.
- Esto permite hacer *Triple Play* (especialmente *VoIP*) a alta *QoS*. También *VPN* entre *Sucursales*. Combina la flexibilidad de las comunic. *E2E* gracias a *IP* y la calidad (*QoS*)/seguridad de *ATM* o *Frame Relay*.
- Esto ha permitido *Integr.* sencillamente redes *Moviles* (*Internet 4G*), terrestres FTTX (*Fibra*) o LAN (*Ethernet*).
- Gracias a esto las *Empresas* han ido *Integr.* sus *Servic.* en *Internet* (a pesar de ser una Red Mejor Esfuerzo *QoS*).
- QUITAR?? Aún así siguen existiendo Redes Propietarias (*Proveedor*) que cuando usaban *ATM* para *Internet* era muy costoso (trabaja con SONET/SDH (*FTTH*)).
- Trabaja con *IGP* (*Rut.*).

Ver Fig. 40.10. *Frontera*

Dada una *Latencia* max. y un A. *Banda* min. = *QoS* ¿que *Rut.* sigo?

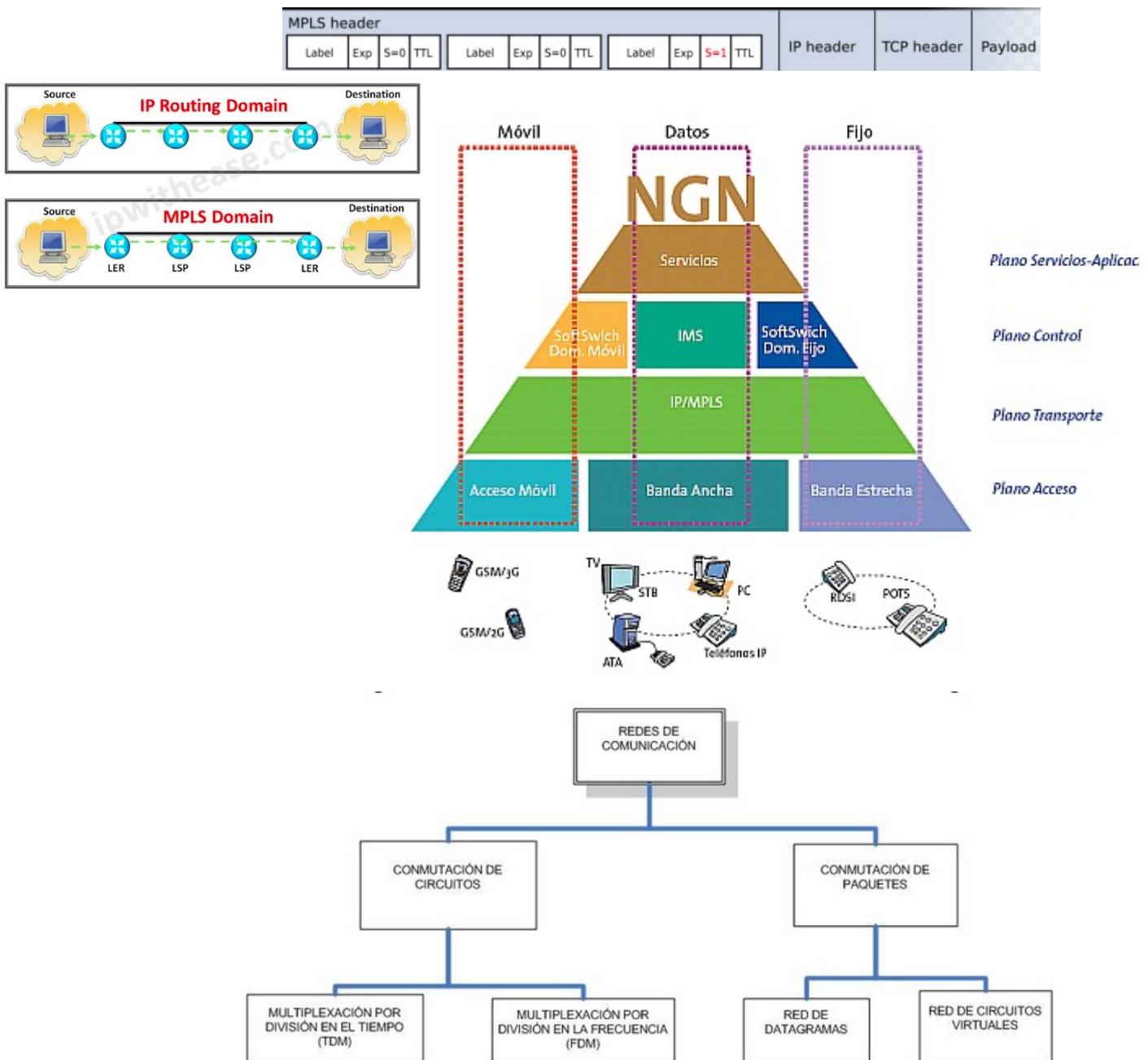
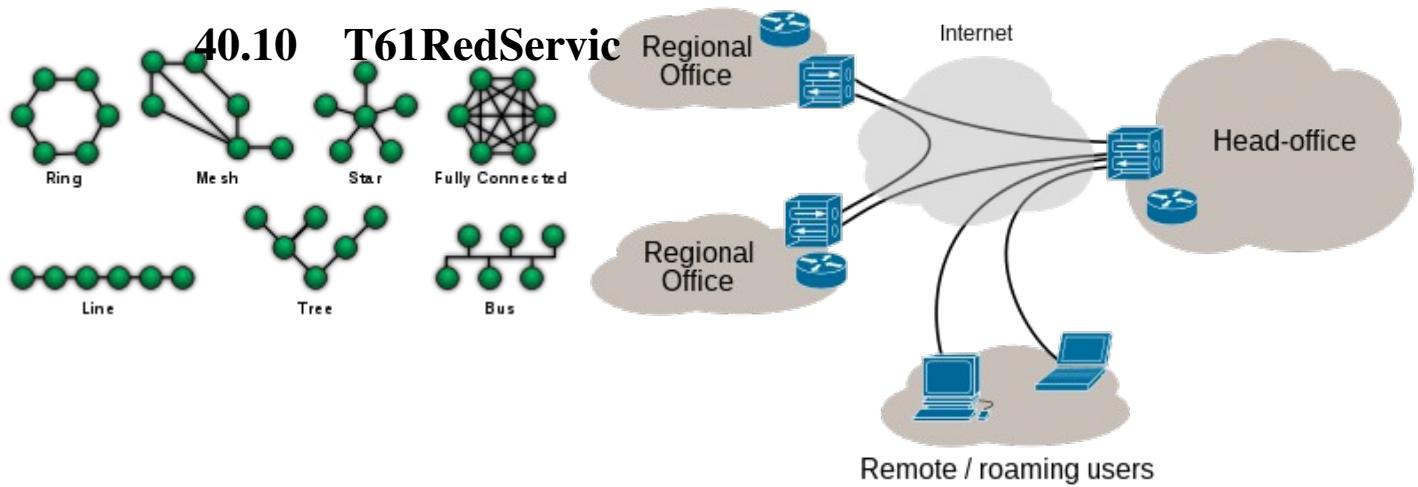
40.8 Otras superadas por MPLS

- **ADSL** (Asymmetric Digital Subscriber Line de *Bell*): *Actual.* *Vel.* bajada 20 Mbps < subida 2.
Reemplazó al RDSI (*Servic. Int.*) y está siendo remplazada por *HFC* por su falta de *Robust.* a *Interf.* pues usa 1 solo *Par Trenz.* que aprovecha el A. *Banda* del Cobre del *Telef.* (aunque *Tier2*, etc tiene Fibra).
Usa *Modulac.* *DMT* y *Modem*.
Mejoras: ADSL2+... Motivado por el *Servic. Int.*. Ver *PPP*, *Proveedor*
- **Servic. Int.** (*Servic. Integr.*): Dos significados
 - a) *Arq.* que contrasta con Serv. Dif. en la que los *Rut.* y *Apl.* implicadas reservan *Recurso* para proporcionar un *Triple Play* de alta *QoS*.

- b) ISDN (Integrated Services Digital Network) o RDSI (Red Digital de Servicios Integrados): red para aprovechar *Telef.* y reemplazada por el *DSL*. RDSI motivó al *DSL* (al definirlo la *ITU-T*). Usaba *PPP*. De las 1^a Redes en ofrecer *Triple Play Voz y Datos* sobre los mismos hilos de RTB *Telef.*. Usa *ATM*, *Cod. Bipolar AMI* y *Frame Relay*, *Cable Estr.* (ISO 11801 *Par Trenz.*).
- **ATM** (ASincr. Transfer Mode, no confundir con Cajero): En desuso por *MPLS* (*Hist. R.* reemplazado por *NGN???* (*QoS, Ethernet WAN???*) aunque está en corazón de SONET/SDH (*FTTH*) y de RDSI (*Servic. Int.*) [*ATM*]. Basada en *Circ. Virt.* Permite *Triple Play* a alta *QoS*.. Pensada en los 80 para satisfacer A. *Banda de Servic. Int.* pero caro por *Modems Cable*. Con *Gateway* se *Interop.* con *TCP/IP*. Ver *PPP*
- **Frame Relay** : usa *Conmut.* Paq. en **Lineas Alquiladas** basada en *Circ. Virt.* (*TCP*) comunic. Originalmente para transm. *Vel.* como *Voz* y *Servic. Int.*. *Hist. R.* En desuso y reemplazado por *VPN* y *MPLS* (aunque en *Sucursales???*)
- **X.** 25: de *ITU*, en desuso. *Hist. R.* por usarlo Red UNO (*PDN*). De los 1º en *Conmut.* Paq. **pero basada en Circ. Virt.**.. para *Transac. Sucursal* con Tarjetas de Credito. Los X. son de ITU: X.500 (*LDAP, Netware*). X.509 (*Certif.*). X.690 ASN.1. Ver *Guerra*
- *Fibra HFC* y *FTTX*.

40.9 Ejemplos de redes

- Red de computadoras: *In/ExTranet, Red Superp., Sucursal, PDN*
- Telecomunicación: *Telef.* (Voz en RT y Dat., Conmut. Circ., RJ11,), *TV* (TV DVB-S), *Movil* (cambio-roaming, RISC-Android, 2G-GMS, 5G, Microondas-OFDM), *Satelite* (GPS, Microondas GEO/MEO/LEO)



Chapter 41

*+62. Arquitecturas de sistemas de comunicaciones. Arquitecturas basadas en niveles. Estándares. (19NovLola)

41.1 Introducción. Conclusión

- Ej-motivador: ¿parecido entre Red y *Empresa* transporte? Sol. *Ud.* y Protoc. =*Modul.* ¿qué protocolo siguen los *Móvil*? Crec. *Moore*. *Robust.* de *Internet*
- Hist. *Guerra* de los Protoc.: SNA *IBM*, IPX/SPX Kahn& Cerf73 DARPA TCP/IP. Zimermann *OSI* Crocker69 RFC ARPANET ICANN antes IANA
- Futuro: *Web3.0 IA, IoT* seguir con *Abstr.*

41.2 Sistemas de comunicaciones. Servicios y servidores.

- Sist. *Comunic.* en *Red*: Nodos comparten *Recursos*
- *Arq.* Red: Fig. 41.10:
 - *Hardw. Red*: a) Terminal (ETD), *Medio*, Comunic. (ECD).
 - b) *Topol.* Red
 - *Softw. Red*: a) *NOS*
 - b) Apl. *Distrib.*: aquí
 - c) *Protoc.*: aquí

41.3 Computación o aplicación distribuida

- *Servid., Host* y *Servic.*
- Comput. *Distrib.* o Apl. *Distrib.*: tenemos las siguientes *Arq.*:
 - *Client.-Servid.*

- MultiTier o N-niveles: *Arq. ANSI/SPARC* y *Dis. Jerarq. Red*
- *Distrib.*: *P2P* o *Cluster*) ver *Flynn*
- Mod. *Client. Servid.*: es un tipo de *Arq. Softw. Distrib.* de 2 *Capas*. No es solo para Redes ej: *Arq. Kernel C. S.* En Redes contrasta con *P2P*. Ej: *Servid. BD*. Ver *Middlew.*, *SO Servid.*, *Comput. Edge (IoT)*, *Master*.
- *P2P* (Peer To Peer, Par A Par, entre Pares o Iguales, no confundir con *E2E*): tipo de *Arq. Red Distrib.* donde los *Nodos* ceden parte de sus *Recursos TAP* (*Almac.*, *A. Banda*, ..) al resto de Nodos sin un *Master* central.
Es una Apl. *Distrib.* que *Part.* la *Carg. Equil.*, implementada como *Red Superp.* (en la *C. Apl.*) en *Internet*.
Todos los Nod. tienen la misma *Pot.* (*SO Ligeros*) y actuan como *Client.* y *Servid.* a la vez.
Tiene *Topol.* Red en Malla (Maq. Boltzmann). Permite *Almac. Distrib.* como *Blockchain* o BitTorrent Ver *MiraCast, IoT*.

41.4 Pila de protocolos

- *Protoc.* (no confundir con *Estand.* (abajo) ni *Prote*): Norma o *Alg.* de *Comunic. Docum.* normalmente en un *RFC*.
 - a) Es *Softw. Red*
 - b) *Interop.*: para Comunic. y/o Compart. *Recurs.* indep. del **Fabricante** los cuales pueden dar distintas *Impl.* del mismo P. Ver estándares *ITU*, *RFC*, *Cript.*, *Integr.*, *Guerra P.*, *Dis. Jerarq. Red*
- Caracter. min. de cualquier *Protoc.* o *Arq. Red*:
 - Acceso al *Medio (MAC)*:
Control Err.
Control de Congest. y *C. Flujo Dat.*
 - *EnRut. (IP)*:
Direc. IP
 - Principio *E2E (TCP)*: *OrCo* (3 pasos *Socket*), *AcRe*, *Sincr. de Paq.*
- *Pila Protoc.* (*Protoc. Stack, Pila P.*): *Arq.* referente de *Red* basada en *Capas*
Casi más pedagógico que real (*OSI*). Cumple las reglas siguientes:
 - a) *Transpar.*: solo se permite Comunic. entre capas del mismo *n*.
 - b) *Servic.*: para *Transm.* usados por capa *n* son solo los de *n – 1* (*VLAN* transgrede)
 - Ej: *TCP/IP* (implement.) y *OSI* (teórico).
 - Es *Filos.* de *Modul.* y *Abstr.*: partir *Cod.* de *Transm.* Juanbe: Si el cod. *OSI* entero es 10MB?? entonces un *Switch* solo C. Enl. 1MB. Es como *Caden. F.* unas se *Llam.* a otras.
 - Similar al transp. mercancías (*PDUD.*) o *Recon. Voz*.

-) *Integr.* Heterogeneidad: subredes como *Internet* se comunic. efic.
 -) Ahorro *Precio*: *Dispos.* solo *Implementan* la C. a usar.
- *Guerra* de los *Protoc.*: *Hist. R.* de 70 a 90 surgieron P. como los de abajo. De los 80-90 *OSI* se peleó con *TCP/IP* y en los 90 ganó este último.
 - **SNA**: *IBM74*, 1º con *Pila Protoc.* y de 7 capas, base de *OSI*. Ver *Gateway*, *SDLC* *PPP*
 - *DECNet*: para conectar 2 *PDP-11* (con *UniBus*).
 - **IPX/SPX** (creado por Novell para su SO *Netware*) o *NetBEUI* (Net *BIOS..* de *Win*) o *AppleTalk* (*Mac*): son ej. *Hist. R.* de *Protoc.* que se usaron en *LAN* y que mantienen algunos hospitales o bancos. Hoy domina *TCP/IP*.
 - En medio salieron *X.25*.

Ver G. *Naveg. Mozilla, Milit.. IPX/SPX*.

41.5 Paquetes y cabeceras

- *Ud.* PDU (Protocol Data Unit, Unidad): en *Comut.* Paq., de forma similar a los Paq. (+*Container*+Barco) en transporte de mercancías o cartas, cuando la información atraviesa una Capa esta se *Encapsul.* quedando el Paq. formado por 3 elementos (Fig. 41.10):
 - *Carg. Util*: Payload o SDU (Servic. Datat Unit). En *TCP* el Max. Segment. Size es 536Bytes ver ej.
 - *Cabecera* : Header o PCI (Protoc. Control Inform.) o Metadatos para su correcto tratamiento: nº orden y punto de llegada. A nivel más bajo más Cabeceras acumuladas (*Pila Protoc.*). Ver F. (*Decl.*), Fuente, Directiva o Macro, Compr. sin pérdidas. Fich. Multimedia, Container, Transgresión VLAN, Monit. SNMP, Diagr. UML (*CASE Umbrella (Cooper.)*), RTP.
 - *Checksum*: del Payload o de ambos.

PDUs famosos:

- *Trama* (Frame): de C. Enl.. En *Ethernet* como Max. es 1500Bytes ver ej. abajo.. Ver *VLAN*, *Colis.*, *Switch*, *Brigde*
- *Paq.* : de C. Internet. Puede Perd. (*QoS*). Ver Rut., *CISCO Packet Tracer*
- *Segment.*: a) de C. Transp.. Si **UDP Datagrama**. b) Dividir un Paq. Dat. cuando es mayor que la Máx. Transm. Ud. (MTU) Ej: 1500B *Ethernet*

Ver *Email*, *Def.* F., Estr. C, *Comut.* Paq. Ud. *Inform.*, Ud. *I/O (Dispos.)*, *Atom.*, *Token*, *Disco HDD o SSD*

ver ej. de MTU de *TCP* y de *Ethernet*

41.6 Modelo OSI

- *Model. OSI* (Open Systems Interconnection, *Sist. Abierto de I.*): publicado en *ISO* 1980 por el francés Zimmermann *Hist. R.* para *Integr. Protoc.* (*TCP/IP*, *IPX/SPX*, *NetBEUI* y *AppleTalk*).
Hoy es la base pedagógica, pero inspiró a reales.
Para poner orden en **fabricantes** de *Hardw. Red* (ver *Integr.*)
Posee 7 capas (Tab. y Fig. 41.10 (FERTSPA)). Ver *Guerra*.

Capa (PDUd.) (Servicios)	Def.	Ej.
<i>C. Fisica</i> (Bit/Símbolo) (Gest. conex. fisica. Constr. transf. bits)	Transmisión y recepción de flujos de bits en bruto a través de un <i>Medio</i> físico (def. la Señal , Cap. <i>Canal</i> , conectores, <i>Cables</i> y dispositivos)	Los 3 <i>Medio</i> Transm., <i>RF</i> , microondas, <i>Serie 232</i> , <i>IEEE 802.3ab</i> ([<i>Pila Protoc.</i>]), <i>SONET/SDH (FTTH)</i>)
<i>C. Enl. (Trama)</i> (No <i>OrCo</i> y No <i>AcRe</i> (R. descarta trama <i>Err.</i>); No <i>OrCo</i> y <i>AcRe</i> ; <i>OrCo</i> y <i>AcRe</i> (si err. resolicta);	Transmisión de tramas de datos entre dos nodos conectados por una capa física (Direc. físico (MAC y LLC subdiv de <i>IEEE</i>), <i>Err.</i> , Control <i>Flujo Datos</i>)	Fast/Gigabit <i>Ethernet</i> y <i>Token Ring (IEEE 802)</i> , <i>PPP</i> , <i>MPLS (2.5)</i> , <i>ARP</i> , <i>FDDI (Fibra)</i> , <i>HDCC</i> . Ver <i>Enl.</i>
<i>C. Red</i> (Paquete) (<i>OrCo</i> (Con/sin <i>AcRe X.25/ATM</i>) y No <i>OrCo</i> (normalmente Sin <i>AcRe IP</i>))	Estructuración y gestión de una red multinodos, incluyendo el Direc. Lógico (IP) , el EnRut. (estat. o dinam.) y el control del tráfico (<i>Congest.</i> de nodos) (en redes de <i>Broadcast</i> esta C. no hace nada)	<i>IP</i> , <i>ICMP (Ping) IGP</i> , (v4 o v6), <i>X.25</i> , <i>ATM</i> , <i>AppleTalk</i>
<i>C. Transp. OSI</i> (Segmento) (<i>Err.</i> , Control Flujo Datos <i>OrCo</i> y <i>AcRe</i> según <i>TCP</i>) y <i>UDP</i>)	Transmisión fiable de segmentos de datos entre puntos de una red (<i>E2E</i>), incluyendo Segmentación (partir info de C. <i>Sesion</i>), acuse de recibo y multiplexación	<i>TCP</i> , <i>UDP</i> , <i>TLS</i>
<i>C. Sesion</i> (Dato) (<i>Autent.</i> entre dispositivos y usr)	Gestionar las sesiones de comunicación, es decir, el intercambio continuo de información en forma de múltiples transmisiones de ida y vuelta entre dos nodos	<i>PPTP (VPN Tunel con PPP)</i> <i>NetBIOS</i> , <i>SSL (TLS??)</i> . <i>RTCP (RTP) Servic. Dir.</i> , <i>Autent.</i>
<i>C. Present.</i> (Dato) (ver Def.)	<i>Traduc.</i> de datos entre un servicio de red y una aplicación (sintax. y semant.); incluyendo la <i>Cod.</i> de caracteres, la <i>Compr.</i> de datos y el <i>Des/EnCrip.. Es el Frontend</i>	<i>ASN.1</i> , <i>MPEG</i> , <i>XML</i> , <i>JSON</i>
<i>C. Apl.</i> OSI (Dato) (dar Serv. de Red a Aplic. fuera de OSI)	APIs de alto nivel, incluyendo la compartición de recursos, el acceso remoto a archivos	Las básicas (ver arriba).

41.7 Protocolo TCP/IP

- **TCP/IP** (Transmision Control Protoc./Internet Protoc.): Implementación del OSI en 4 capas según [Stallings07] Fig.41.10).
 - Principio *E2E* o *Filos.* parches de *Internet:* **IP no garantiza entrega y TCP si (AcRe)**
Sus Protoc. se han ido *Estand.* con el tiempo por *IETF-RFCs*, ej. no es *RT* pero *RTP*
 - *Hist. R.* del DARPA [Kahn& Cerf73] (*Turing*) para ARPANet (ver *RFC*), la 1^a Red *Exten.* (*WAN*) con *Commut.* Paq. y precursora de *Internet*, entre comput. *Unix* (luego se extendió a *Win*). Ver Gusano Morris 1º *Malw..*
 - Ganador en *Guerra Protoc*

Es *Unif.* por *E2E*

Capa (PDUs)	Def. (Servicios)	Ej. (los de OSI correspond.)
C. Acceso a Red [Stallings07] o Enlace (link)	solo especifica que el protocol. debe permitir el S/R (Envio/Recepc.) de Paq. entre el <i>Host</i> y la <i>Red</i> (si viene una nueva <i>Tecn.</i> como la <i>Fibra</i> su estandar debe especificar como). A pesar de todo <i>CISCO</i> la separa en <i>C. Fisica</i> y <i>C. Enl..</i> Contiene todos los <i>Host Acces.</i> dentro de la <i>LAN</i> (sin pasar por el <i>Rut.</i> , ver <i>Red Superp.</i>).	
C. Internet (Paq.)	(servicio solo de <i>Commut.</i> Paq. No <i>OrCo</i> , para EnRut. y evitar <i>Congest.</i>)	
C. Transp. (Segmento)	Las de <i>TCP</i> y <i>UDP</i>	
C. Apl.	La experiencia ha demostrado que agrupar las 3 últimas C. OSI es lo mejor. (Ofrece Serv. de Red a Usr/Apl)	

Como se *Transm.* (o recib.) un Dat.: se va encapsulando, a nivel más bajo más *Cabeceras* acumuladas. Fig. 41.10: Apl. del Emisor → C. Transp. (*Puerto*) → C. Red (*IP*) → sale del *Host* y pasa al *Medio* donde sube y baja → Receptor → camino *Inv..*

41.8 Comparación OSI con TCP/IP

- **TCP/IP vs OSI** u OSI vs TCP/IP: en la Tabla observar que el *Ambito* (hay quien habla de softw. o hardw.) del Host es hasta C. Transp. (ver *Pila Protoc.*). Hay modelos híbridos.

<i>TCP/IP (ej. Protoc.)</i>	<i>OSI</i>	<i>Ambito Capa</i>	<i>PDUd. (MTU)</i>
<i>C. Acceso (Ethernet, ARP)</i>	<i>C. Fisica y C. Enl.</i>	<i>Canal, Medio</i>	Bit, <i>Trama (< 1500B)</i>
<i>MPLS (ambas) VLAN (transgres.)</i>			
<i>C. Internet (NAT, IGP, MPLS)</i>	<i>C. Red</i>	<i>Medio</i>	<i>Paq.</i>
<i>C. Transp. (UDP)</i>	<i>C. Transp. OSI</i>	<i>Host</i>	<i>Segment. (Datagram.) (< 576B)</i>
<i>C. Apl. (DNS)</i>	<i>C. Sesion, C. Present. y C. Apl. OSI</i>	<i>Host</i>	<i>Dat.</i>

- Similitudes: ambos son *Pila Protoc.* y Servic. de Capas similares
- Diferencias: 7 vs 4 Capas
- Fracaso de OSI: no ha llegado a *Impl.ementarse* y solo se ha usado teóricamente por:
 - Teórico y Complicado: tantos niveles (sobre todo 5, 6 y 7)
 - Tarde: salió cuando TCP/IP llevaba años funcionando y estaba extendido en EEUU (el mayor consumidor)
 - Controlador: en los 80 la UE solo permitía este
 - Prototipos: de equipos malos y caros
 - No Intercapa: no permite que capa N use servicios de capa $N - 2$ directamente. TCP/IP si permite a Apl. usar IP o MAC.

41.9 Organizaciones de estándares

- Organz. *Estand.* o Norma (no confundir con *Protoc.* ver abajo): *IEEE,...* definen las Normas (*Syntax.*) de *Impl.* para *Interop.* y para una tarea *Tecn.* repetitiva indep. del fabricante. Las de TeleComunic. nacen debido al crecimiento vertiginoso de las *Redes* para establecer *Protoc.* (intercambio de informac. indep. del fabricante y *Proveedor*).
 - E. Facto: aquel no concensuado de forma oficial sino por su *Popularidad* Ej: *SQL, Arq. ANSI/SPARC, IEEE 802, x86*. Ver *Arq.* ref.
 - *Protoc.* (*RFC* y *Comunic.*) vs *Estand.* (hay de redes, de edificios, son *Tecn....*Ej: *Wifi*).
 - * Ej: *Estand.:* *Cod. ASCII+Compr.+EnCrip.*
 - Ej: *Protoc. TCP/IP vs OSI.*

Ver *Metr. Softw., Benchmark, Music. MIDI, Std.*

- **ITU** (International Telecommunication Union): consta de ITU-T (teleco, *JPG, Servic. Int.*), -R (*RF*, parte la *RF*, Ej: ISM del *Bluetooth*) y -D (desarrollo, acceso sostenible a las teleco). Ej: X.25 (*Servic. Int.*), V.90 (*Modem*) y *Frame Relay*, G.657 (*Fibra Colibrí*). G.711 *Cuantiz. Voz*
- **ISO** (International Standard Organization): de normas industriales y comerciales. Por países en España AENOR, EEUU ANSI. Ej: C, SQL, JPG, MP4 y MP3 del M/JPEG, *Imag. Disco* 9660 (ver). ISO/IEC 11801 (*Par Trenz.*), ISO/IEC 80000-13 *Kibibit* (Ud. *Inform.*), *Unicode*=ISO/IEC 10646
 - IEC (International *Electr.onical* Commission): trabaja mucho junto a ISO
- Reguladores de *Internet*
 - **IETF** (Internet Engineering Task Force, *Grup.* de Trabajo de Ingeniería de Internet, no confundir con W3C para Webs): responsable de todos los Estand. del Protoc. *TCP/IP*, pues regula los *RFCs*. Se coordina con ICANN (*CIDR*) y es financiada por la ISoc con 1 millón dolares/año. Ver *MPLS, CIDR, Unicode UTF – 8, SNMP (Monit.), NFS*
 - **RFC** (Request For Comments, *Petic.* De Comentarios): memorandos (en *Texto Plano*) regulados y puestos en la *Web* del *IETF* por expertos para que otros lo revisen. Nacen con Crocker69 para *ARPANET Hist. R.* Ej. de RFC: los *Protoc.* más importantes de *Internet* (IP, HTTP, RTP, TLS, también LDAP). Ver Red Priv..
 - **ICANN** (Internet Corporation for Assigned Names and Numbers): Org. *Estand.* para Asign. DNS (*Domin.*) e IP y mantener su *BD*. Antes F. del IANA (Internet Assigned Numbers Authority, ej. *CIDR*). *Hist. R.* Se coordina con el *IETF* como
 - **ISoc** : la única ONG sin ánimo de lucro que financia *IETF* para los *RFC*. Promueve el uso mundial de *Internet*. También ha creado Internet Architecture Board (IAB, aprueba normas), Internet Engineering Steering Group (IESG).
 - **W3C** (World Wide Web Consortium, no confundir con *IETF* que es para IP): Estand. la *Web*. Ej: VoiceXML
- **IEEE** : ha desarrollado la familia de estándares IEEE 802 (*Ethernet, ...*). Otros IEEE 754, *POSIX*
- **ANSI** : American National Standards Institute. Representa a EEUU en la *ITU* e *ISO* que luego adaptan muchos de sus estándares. Ej. C, SQL, ASCII. La **TIA** (Telecommunication Industry Assosation) normaliza al *Par Trenz.* T568. Arq. ANSI/SPARC, Simbol. *Diagr. Flujo*.
- Otras:
 - EIA (*Electr.onic Industries Association*): cesó en 2011. Ej: el Serie 232, T568 (*MDI Trenzado*)
 - **ETSI** (*Europ. Telecommunications Standards Institute*): no confundir con Escuela Técnica Superior de Ingenieria. Ej. GSM Movil y *RFC* del RTP
 - JEDEC (Joint Electron Device Engineering Counci): ej: GDDR SDRAM
 - OMG (Object Management Group): *Diagr.* UML, Obj. Distrib. (*Progr. Compo.*).
 - NIST (National Institute of Standards and Technology): ver AES, Estand. Segur.

41 – *+62. Arquitecturas de sistemas de comunicaciones. Arquitecturas basadas en niveles.

Estándares. (19NovLola)

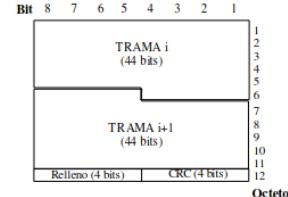
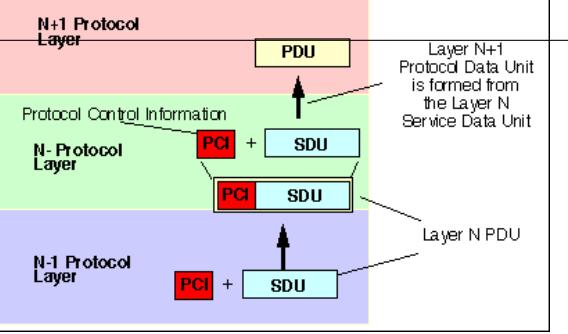
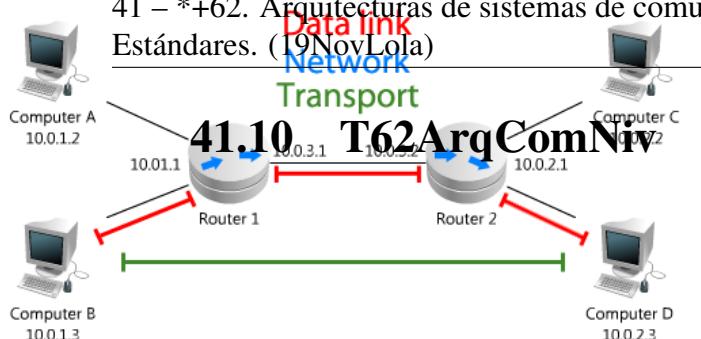
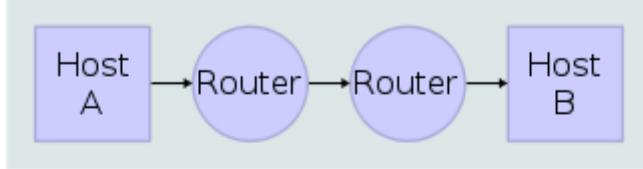


Figura 5. Formato de un *frame-pair*.

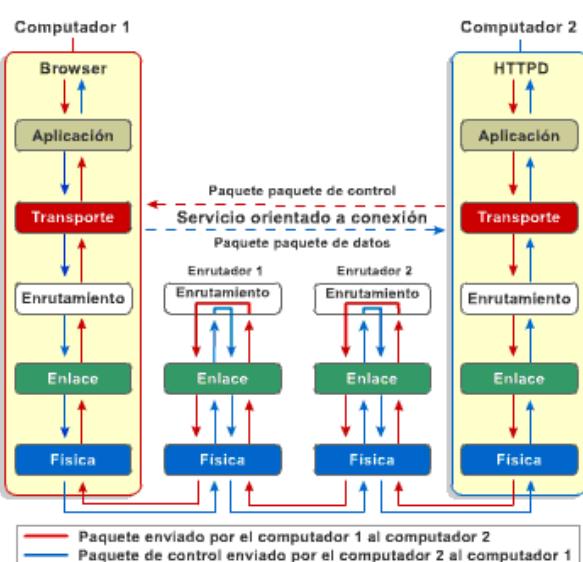
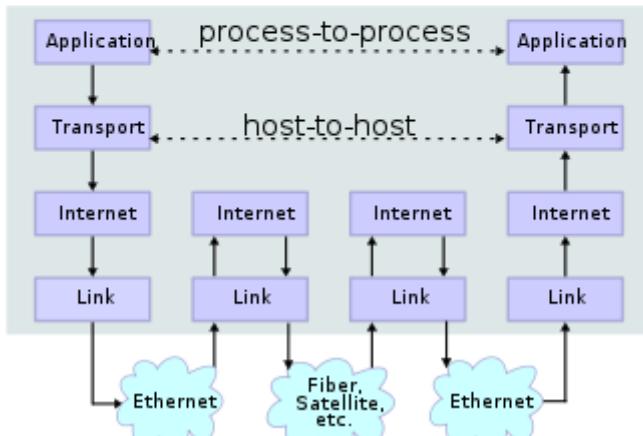


Figura 6. Encapsulamiento de un *frame-pair* vía RTP.

Network Topology



Data Flow



TCP/IP model	Protocols and services	OSI model
Application	HTTP, FTP, Telnet, NTP, DHCP, PING	Application
Transport	TCP, UDP	Presentation
Network	IP, ARP, ICMP, IGMP	Session
Network Interface	Ethernet	Transport
		Network
		Data Link
		Physical

Chapter 42

+*63. Funciones y servicios del nivel físico. Tipos y medios de transmisión. Adaptación al medio de transmisión. Limitaciones a la transmisión. Estándares. (8DicJuanA)

42.1 Introducción. Conclusión

- Ej. motivador: en los 20 Nyquist dijo que por un *Telegrafo* la Vel. de *Puls.* PS es $< 2B$
¿Por qué? ¿Porqué no podemos doblar los *Cables Película canadiense-belga Proyec.* Colibrí 2019.? ¿Por qué *Vel.* baja en largas distancias como Marte? Sol: *Shannon*.
¹ Crec. mas que *Moore* en Redes
- Importancia de la *Vel.*: **Bondades** de las redes: para *Tele* trabajo (zona rurales) y Dat. *Masiv.* crecimiento exponencial que llega a superar L. *Moore* [WikiFibra]²
- Vamos a conocer los límites en Transm. que establecen los estándares de calidad en vel./*Precio*: OJO: **Nos centramos** en *Red* larga dist. y entre *Comput.*: *Pila Protoc.*, no *Bus* o *Jack* audio Pendrive o video tranm. info.
- *Hist. R.*: - Fourier (termico, *Transf.*), Maxwell, Marconi - Heaviside *Coaxial*, - *Bell Par Trenz.*, - Nyquist *Tele*, - *Shannon* - Kao, Kapany60 *Fibra*
- Conclusión: Tma *Shannon*
- Futuro: Inform. *Cuant.* (No *Intercep.*), *USB-c+Ethernet*
- Otras conclusiones:
 - La idea es antigua (Heaviside...) solo mejoran materiales y ancho *Banda* \Rightarrow por *Shannon Vel.* (basta ver tabla *Ethernet*)

¹Ej. Motivador: ¿Vel. máxima de *Puls.* por Fibra o *Par Trenz.*? ¿Mejor medio barato y robusto (lejos y sin errores)? ¿Como va la señal en *Cable Ethernet*?

²Importancia: *Comando apt*, *Videojuegos*, *TV a demanda...*

- *Err.* Concealment *Perd.* *Paq.* *QoS* (bitrate) autoadaptarse
- *IoT* y *Embeb.*, Protoc. CAN Bosch de Automoción

42.2 Sistemas en Red

- *Red Tele.*
- *Pila Protoc.* *OSI* vs *TCP/IP*

42.3 Funciones y servicios del nivel físico OSI

- *C. Física* : Nivel 1 del *OSI* que se encarga de tomar la *Trama* del N2 (*C. Enl.*) del Emisor y de **Adaptarla** (ver) al *Medio Fisic.* (*Medio Transm. ∈ Hardw. Red*) de forma que el N2 reciba *Transpar.* la misma trama. Su *PDUs*. es el bit. Proporciona y *Estand.* los medios *Fisic.* mecánicos (*Fibra*), eléctricos (Volt. *Imped.* max) y de *Temporalizar* (*Signal*) para *Transm.* Inform.
- RESUMEN: Las principales F. y *Servic.* de C. F. son: aparte de transm. sin *Err.* y *Transpar.* a) Def. *Signal* (*Modul.*) b) *Medio* (*Conect.+Fibra*) c) *Topol.* Red: como el *Hub* (aunque aquí no profundizamos) **IMPORTANTE**
- *CISCO*: si la distingue aunque *TCP/IP* no (en *C. Acceso*). Su *PDUs*. es el bit o *Signal*.

42.4 Fundamentos: Nyquist, Fourier, ancho de banda, canal, Shannon-Hartley

- Nyquist. *Equiv. Analog. Digit.*
- *Transf.* Fourier Discr. (*Digit.*): Comb. *Lin.* de expon. \mathbb{C} del mismo nº de muestras, duplicada en espejo \Rightarrow *Cod.* magnitud+fase. *Pot.=suma*² por Parseval. Reconstr. con menos bins ok si *PCA*
 - T. Fourier *Analog.* *F*: debe ser $L_{p=2}$ integrable. *Equiv. Analog. Digit. (Muestr.). Puls.* $F(\text{sinc}(t)) = \text{cuadrado}(f)$. Posibilidades: *Banda/Energ.* finito/infinito a/periódica no/ventana.
 - FFT: es el *Alg.* Num. más importante y paradigma de *Efic.* ($\text{BigO}(N \log N)$ vs $O(N^2)$, se desconoce si puede ser menor *NP*). Se hizo para terremotos y detectar Bombas Atómicas por Tukey65 *Hist. L.* aunque Gauss para Interpol. (supongo vió el *Paral./Repet.* de coef.). Ej: DeModulac. OFDM 5G Movil, Filtr. MATLAB, multipl. Array Estr. tipo Toeplitz..). Ver *DSP*

Ver *NP*, T. Aprend. Autom., Modulac., Lista Circ., FFT (GPU, Div. Venc.). Corr., PCA, Est.

- **Canal** de *Comunic.* (no confundir con *Medio ni Medio Transm.*): puede ser Fisic. (*Medio Transm.*) o Log por *Multiplex*. [WikiComunChannel]. Yo lo entenderé como Log. por Cap. C. *Shannon* o por C. de *RF de Arduino* (1, 2,...).
 - **Cap. Canal** (Tma *Shannon-Hartley* $\in 2^a$ Tma *Shannon* o Tma *Cod.* de *Canal*): $C = W \log_2(1 + S/N)$ ver Ej. *Par Trenz.. Dem.* basada en *Vel. Muestr.* [Nyquist28] (*Hist. R.*): “la freq. max. de *Puls.* por un *Telegrafo* debe ser menor que doble del A. Banda $f_p < 2B$ ” y en la *Equiv. Analog. Digit..* NOTA: **A más dist. más Aten. y menos Vel.** Ej: antijammin *Wifi* de Lamarr.
 - **Banda** : varios significados para a) A. B.: rango freq. de un *Canal* o *Signal*, se asemeja a montículo con zona plana a $= 0dB$ y caídas $(f_{min}, f_{max}) = -3dB$ b) A. B.: = *Vel. Transm.* (bps) por *Shannon*, interesa aprovecharlo al máximo. c) B. Base/Pasante: ver *Modulac., Cod. Lin.* Ver *QoS, DSL, OFDM, Modem, Bus, Mem., Antena, Sens.*

4 caract. *Medio Transm.* del Tma (ver). Ver C. *I/O, RAM MultiC. Fibra, Fulkerson.* Capacidad *Almac., HBM (RAM 3D)*

42.5 Comunicación. Transmisión. Medio de Transmisión

- *Comunic.:* Sim/Semi/*Duplex* y *A/Sincr.*
- **Transm.** : es enviar *Dat.s* en una dirección por un *Canal*. Lo contrario es R. (Recibir). Aunque puede ser *Duplex*, no confundir con *Comunic.* que es enviar y recibir. La *Vel. Transm.* se mide en bps (*PS*) y se suele T. en *Serie*. No es solo por radio, si no ej. de Pendrive a PC. Ver *TAP, Compr.* Distinguimos según:
 - *Analog.* o *Digit.* según *Signal*
 - *Serie* vs *Paral.*
 - Las 2 de *Comunic.:* *Duplex* y *Sincr.*
 - *Latencia* o *Vel. Propagación* (no confundir con Cap. *Canal*)
 - *Topol.* Red vs *E2E (Serie 232)*
- *Metr. Transm.:* relacionadas con T. *Shannon (Canal)*
 - Cap. *Canal*
 - A. *Banda= Vel.*
 - Aten. uación *Pot.=Dist.=espacio entre Repet.* (dB/m Ley Beer-Lambert)=*Ext. Red*
 - Prob. *Err.=Robust.* (*Apantall.*)
- **Medio** Transm. (no confundir con C. *Canal ni Medio Comunic.*): *Esp. Fisic.* que *Transm.* o propaga una *Signal* para *TeleComunic.* (\in *Hardw. Red*). Es un tipo de *Canal Comunic..*
 - *Wireless/Cable* o No/Guiado o In/alambrico: los 1+3 son: *Wifi+Par Trenz. Coaxial Fibra*. Otros aquí no: *Perif. Jack, Altavoces (luz), Pendrive a HD.*
 - *Imped.:* *Filtr. RLC* la señal
 - *No/Lin.:* si permite superposición Comb. L. de ondas.

- Costo y Facilidad *Instal.*
- Otros 4+4+2 *Canal+Transm.+Topol.* Red: *Vel., Banda, Aten. Robust. y Transm.*: 4 (*Analog., Serie, Duplex, A/Sincr.* +2 (*Topol. Red, Latencia*)

Ver *Hardw. Red, Cable Estr.*

42.6 Medios de transmisión guiados: los 3 tipos

- **Par Trenz.** (Twisted/Braid pair): el más común. *Cable* de cobre con aislante y entrecruzados en hélice (Fig. 42.12) que anulan *Interfer.* y *Diafon.* externas (idea de [Bell1881] *Hist. R.*). ³. Normalmente hay 4 pares dentro de una manguera (pero también 2 o 25 pares).
 - **Apantall.** (Shielding): = jaula de Faraday (conectada a *Tierra*). Ej: A. *Coaxial* y del UTP vs STP *Par Trenz..* Solución a *Diafon..* Ver *EM, Electr.*
 - Tipos: Fig. 42.12 *U Unshielded* (desprotegido, sin forro en el par), *F Foiled* (forraje externo antes del / o por pares en nombre), *S Screen braid* (pantalla trenzada) U/UTP (*ISO*) o UTP (*Unshielded (NoApantall.) Twisted Pair*, abreviación industrial): sin forraje y menos *Robust.*, *Imped.* de 100Ω (Ohm), con *RJ45* normalmente para *Ethernet*.
F/UTP o FTP (Foiled twisted pair): con forraje global a todo el *Cable* de aluminio, conectado a *Tierra* del PC para eliminar tensiones residuales, 120Ω .
U/FTP o STP (Shielded twisted pair, no confundir con A. Recubr.): con forraje por par, menos flexible y caro, 150Ω , con *RJ49*, solo en entorno muy *Ruidosos*).
F/FTP o FFTP: con forraje globa y por par, mas caro
S/FTP o SSTP: con forraje trenza o malla (screen braid) y por par.
 - *Cable Estr.* o ISO/IEC 11801
 - Categorías: hay CAT1 (1Mbps *Canal*=1MHz) a CAT8 (40Gbps, 2000MHz) y todos a 100m (Fig. 42.12). Más usadas de CAT5e (1Gbps, 100 – 250 kHz, ver problema) a CAT7 (10Gbps) (ver Fast/GigaEthernet)
 - **MDI** (Medium Depend. Interface): *Estand.* más comun es el TIA/EIA-568 (de ANSI) que define **Directo T568A** (RECUERDA comienza en verdes y termina en marrón, conecta dos Dispositivos del mismo tipo (PC a PC), Fig. 42.12) y **Cruzado T568B** (RECUERDA comienza con naranjas y termina en marrón, dos distintos (PC a *Switch*)). El Auto-MDI-X permite ej. que mi TP-Link Switch elija el tipo y poder conectar switches sin *Cable* cruzado (que seria lo habitual).
 - **RJ 45:** es el *Conect.* por excelencia (no confundir con el RJ11 de Red *Telef.*). Llegó sobre el 95, antes Conect. *Coaxial.* RJ49 es blindado con toma *Tierra* para *Apantall.* (ver STP). Ver *Herram. Instal.*

Si el *Par Trenz.* CAT5 tiene Ancho *Banda W* = 100 MHz (ver) y enviamos a una *SNR* (*Pot.* a 100 m supongo) de $S/N = 1000 (= 10 * \log_{10} 10^3 = 30 \text{ dBs})$ ¿cuál es la *Vel. max.* de transmisión: $C = W \log_2(1 + S/N) = 100 \log_2(1 + 1000) = 1 \text{ GbPS}$. Este

³No entiendo la *Fisic..* Las Trenzas forman un *Grup.* $t1 * t4 = t0$

es su límite teórico, lo recomendable es trasmitir a 1/10 i.e. a 100 Mbps. Supone *Ruido gauss*, si ruido rosa $C = \int_0^W \log_2(1 + S(f)/N(f))df$.

- **Coaxial** : conductor de cobre + aislante + *Apantall.* metálico (conectado a *Tierra* para evitar *Interfer.*) + aislante [Heaviside1880] *Hist. R..*
 - C. *Banda Base* (Grueso y Fino): *Imped.* de 50Ω usado en *LAN* C. *Banda Ancha:* 75Ω para *TV*
 - **HFC** (Hybrid Fiber Coaxial): mezcla *Fibra* (entre Nodos zonales, fuera de *Home*) y Coaxial (desde Nodo zonal a casa) gracias a *Cable Modem* (*PC* → Coaxial o Fibra). Usado por *Proveedores de TV* desde los 90, el 1º en permitir *Triple Play*
 - *Imped.* C.: [Getty601] $C = 2\pi\epsilon_0 \frac{L}{\log(R_b/R_a)}$ (a mas *L* más *C*). Para Condensador cuadrado: $C = \epsilon_0 \frac{A}{d}$
 - *Conect.:* **BNC** es una vs miniatura del famoso Conector C que es un tipo de Conect. de *RF*
- **Fibra** Opt.: creada por [japones Kao o indio Kapany60] *Hist. R.*
 - Nucleo vidrio-silicio-plástico: por donde van *Puls.s de Luz* (normalmente Digit. aunque a veces Analog.) Fig 42.12
 - Revestimiento (cladding): vidrio con otro **índice Refrac.**
 - Plástico (antiinclemencias).

Permite gran *Vel. Transm.* y *Ext. Red.* Debe *Instal.* recto (sin rebotes, [ProyectoColibri]). A. *Banda Shannon??*⁴

- Single-Mode Fiber (SMF, Unimodo o Monomodo): más cara. Dist. Largas 5km-40km según *Ethernet*. Nucleo fino, Luz laser Fig. 42.12. Más común *Multiplex.* WDM que Multim. *ITU G.657* (dobléz <3 cm). OJO: **IMPORTANTE** común en latiguillos (*SC*)
- MultiMode Fiber(MMF, Multimodo): más barata. Cortas 500m-2kmm (1Gbps, edif, campus,..FTTH??). Nucleo gordo, Luz diodo. *Distors.* Modal y Cromática da Dispers. *ITU G.651.1*
- *Conect.:* **SC** (Standard Connector) y **LC** (Lucent Connector), ambos *Populares* con *Vers.* para SMF o MMF, vienen en los latiguillos caseros (FTTH) SMF. Otros: **FC** y **ST** (antiguo, para Multimodo) Fig. 42.12. La Férrula es la punta blanca y la mejor es de cerámica. Conectan a Fotodiodos y *Sens.*
- **FTTH:** cortar perfecto G.657. GPON, SONET.
- FDDI (Fiber Distributed Data Interface): para *LAN* basado en *Token Ring* y reemplazado por *Ethernet* (ver).
- Otros: *ISO/IEC 11801* (ver), SONET/SDH (FTTH). *Conect.* TOSLINK (para *Audio*)
- *Multiplex.* Div. long. Onda (WDM): en una sola *Fibra* lasers de difente color (gracias a *Lin.??*). Es un tipo *Modulac..* Ej: con 2 canales podemos tener comunic. *Duplex*, pero se llega a 160 canales, normalmente en **SMF**. Esto junto a **Ampl.**

⁴por freq. luz alta ⇒ A. Banda alto y supongo *Vel. Puls.* limitada a *Muestr.* Nyquist.

Optica (*Repet.* sin electrónica!) despegó las **Redes Opticas** (4a Gen. duplicando *Vel.* cada 6 meses desde el 92 al 01 10Tbps (más que L. Moore!). El record de cualquier *Comunic.* lo tienen unos Japoneses en Nov. 2021, 320 Tbps a 3000Km *Hist. R..*). Usado en SONET/SDH (*FTTH*).

- *Cable MAREA*: de *Proveedor Telef.onica/Amazon/Facebook*, entre EEUU y Bilbao, *Tier1* y *Cable Submarino*, tiene *Vel.* de 26 Tbps (para las 8 fibras del *Cable*). Llega a 40 Km, e implementa *Repet.* de 20dB para aumentar a 60 dB (luego 40 dB es lo mínimo tolerado). España es referente *Despl.* de F. Permite
- *Segur.:* no *Intercep.* o pinchable (mito según Fiber Tapping)

Ver *PPP, HFC, Pullback, Join*

	<i>Vel. /Dist./Ext. Red</i>	<i>Aten. / Robust.</i>	<i>Precio/Instal.</i>	<i>Ej. Analog./Digit.</i>
<i>Par Trenz.</i>	1Gbps (CAT5e alante Giga-bitEthernet y Apan-tall.)/50m (todas CAT)/LAN Ethernet	alta (poner <i>Repet.</i>)/baja (ADSL)	barato/flexible-fácil	A./D., TV anal./digit, <i>Telef.</i> , ADSL
<i>Coaxial</i>	100Mbps/500m?? / más que Par T. (más que Par T.)/ LANMAN		caro / compleja (rigido)	A./D., TV por <i>Cable</i> , Internet mixto <i>HFC</i> o <i>FTTH</i> (<i>Fibra</i>), primeros <i>Ethernet</i>
<i>Fibra</i>	100Mbps/2km Multim. ⁵ vs 40/3000km (Monom. MAREA/- Japón)/MAN y WAN	Baja/alta	Cara (por <i>Conect.</i>)/ligera pero complicada por fragilidad (ver FTTH)	Sobre todo D. <i>Triple Play</i> (voz, internet, video), <i>Tier1 Cable Submarino</i>

- RESUMEN Distancias medias: Par Trenz. 0.1km, Coax. 1km, Fibra 10km, Radio 100km

42.7 Medios de transmisión: no guiados (Wifi)

- *Medio Transm. Wireless o No Guiado:* por vacío o aire = libre=EM
 - *Direc.cional:* haz de *Antena* emisora alineado con *Antena receptor*a
 - *Omnidirec.:* = *Broadcast* perdida natural de Intensidad (*Pot.*)
- *RF* (Radio Frecuencia) ondas del *EM* entre 1m (1GHz)y 1km (MHz). Puede ser de ondas terrestres o satelite:

⁵o 10Gbps/500m por *Shannon* según 100BASE-FX *Ethernet*

- ISM (Industrial, Scientific and Medical): porción del espectro establecida por la ITU-R para fines ISM, no de Telecom. Ej: *Bluetooth, WiFi* a 2.4GHz
- RFID: las pegatinas pasivas que *Id.* productos en Supermercados
- DAB: ver *TV Digital Terrestre*.

Ver *Electr.*

- *Lifi* (Light Fidelity): *Wireless* bidirec. que transm. datos por *Luz led o Infrarrojo* donde el *EM* no posible como en cabinas aviones

	Vel./Dist.	Otras	Ej. uso
<i>RF</i>	3kHz-1Ghz/larga (extraterrestre!)	no Robust., Omnidir.	Radio, TV, WiFi (2.4 o 5 GHz), <i>Bluetooth??</i>
<i>Infrarrojo</i>	<16Mbps (baja)/corta (JamesWebb)	Unidir. 20m	Mando TV o proyector, <i>Perif.</i>
<i>Microondas terrestre</i>	/larga	no Robust. (al clima), Unidir. (Ant. parabol.), mas frec. más Aten.??	Estac. base <i>Movil</i> (5GHz, M. bajas), WiFi (2.412 GHz), Comida (2.472GHz), TV-TDT, LAN innal. <i>WiMAX</i>
<i>Microondas Satelite</i>	/larga	Omnidir. Retardos de Transm.	TV (<i>Broadcast</i> a muchos), Satelit. GEO (órbita geoestacionaria), MEO (media), LEO (baja), Telef. y redes distant.

42.8 Técnicas de adaptación: codificación y modulación

- Adapt.: *Transf.* la *Signal* a *Transm.* (de forma *Transpar.* en *C. Fisica*) para que llegue OK al receptor. Las más comunes son:
 - a) Crecer *Pot.* y *Modul.* (*Multiplex.*)
 - b) Conv. *Anal./Digit..*
 - c) *Cod.*: para *Interop.* Otras ver Sol. en *Distors.*
 - d) *Transcep.* Badum **IMPORTANTE**
 - Otras A.: *QoS* (bitrate en *C. Apl.*)
- *Cod.* en *Lin.*: C. en *Banda Base* o sin *Modul.* $\{0, 1\}$ o $\{-1, 1\}$ de Volt.
 - NRZ (Non Return to Zero): el clásico $1 = High Volt$ $0 = Low Volt$. Requiere la mitad de A. *Banda* que *Manchester* pero no es C. *Sincr. AutoReloj* (requiere reloj aparte).

- **Manchester** : según el IEEE 802.3 (Ethernet viejo) $1 = 0 \rightarrow 1$ Flanco de subida, $0 = 1 \rightarrow 0$ F. bajada Fig. ???. Ocupa doble A. Banda que NRZ pero es **C. Sincr.** **AutoReloj** [Self-clocking signal]. Mejora su Efic. con $8b/10b$ (HDMI) y $64b/66b$ (Ethernet moderno). Ver *Supercomput.*
- Otro: Manchester Diferencial o Biphasic mark code (CC), : Reloj combinado con Dat. para formar una señal Sincr. AutoReloj. Usado en Token Ring
- Otro: BiPolar NRZ: AMI (Alternate Mark Inversion) (en Servic. Int. RDSI). Unipolar: Bifase-L.
- **Modulac.** : es una forma de Cod. donde I: Signal de Dat. (Banda Base, ver abajo) + portadora c , O: s. modulada (Banda Pasante). Fig. 42.12. Variantes usadas en Movil, Modem, TV o radio.
 - Si portadora Analog.: AM, FM, ASK, FSK o QAM (Multiplex. OFDM)
 - Si p. Digit.: PAM, PWM (Puls., Arduino) o PCM (1 Byte Fibra). Ej: M. Espectro Ensanchado de Lamarr (Wifi).
 - M. Delta: se sobreMuestr. a unas varias veces la f Nyquist (es Compr. Diferencial)

Otras:

- **Banda Base/Pasante:** rango de f. de una Signal O. de un Transduc. y que no ha sufrido Modulac.. $[0, f]$ /rango de la modulada $[c - f, c + f]$.
- PLL (Phase Lock Loop, Lazo de Enganche de Fase): para DeModulac. de FM
- **Multiplex.** en Frec.: forma de aprovechar el A. Banda por Comb. Lin. de ondas EM Ortog..
 - **OFDM** (Ortogonal Frequency-Division Multiplex.): es un tipo de Modulac. en F.+QAM o PSK. En muchos Estand.: Movil (4/5G, DeM. con Transf. FFT), TV TDT (DVB-T, Broadcast), ADSL (DMT, Discrete Multi-tone Transmission, version Cableada del OFDM)
 - WDM (muy importante en Fibra)

Ver Ej: PLC Red , Modem.

42.9 Limitaciones a la transmisión

- **Distors.** : alteración o Transf. (T) de una Signal Transm. por Canal. Ej: ver Aten. Aunque la T. puede ser cualquiera ($\text{Alg. } y(t) = T(x(\vec{t}))$) se suele poder descomponer así: $y(t) = h(t)*x(t) + n(t) \iff Y(f) = H(f)\dot{X}(f) + N(f)$ (Recurs. y Lin. Fig. 3.11), donde h es la D. interna por el Medio y n la externa por el Ruidos.
- Soluciones: gracias a la Redund. y al PCA podemos recuperar bajo ciertos límites parte de la Signal original. Ej: bins enMasc. por el ruido son irrecuperables.
 - Transf. o Modif. Espectral: $Y(f) = H(f)\dot{X}(f)$ - FDM aprovechar Banda si sobra.
 - Tecn. de Conformación Puls.os: limitan A. Banda ocupado como $sinc(t) = \sin(t)/t$ cuya Transf. es un rectángulo y viceversa $F[sinc(t)] = rect(f)$. Ver Fibra

- Equalizar después o *Filtr.* antes *Signal PCA*⁶. - *Modul.*⁷ - Reducir *Vel.*⁸
- *Aten.* de *Pot.*: $y(t) = x(t)/A$. Sol: aumentar *Pot.*
- Retardo o dispersión (ver abajo): $y(t) = x(t - r)$. Sol.: - *Filtr.* fase. - Cambiar f. portadora
- *Ruido* aditivo: $y(t) = x(t) + n(t)$
- *Ruido* : tipo de *Distors.* normalmente aditiva con Mod. *Est.. SNR*: *Signal to Noise* relac. *Pot.. 3dB* límite
 - Térmico: agitación de e^- . Ej: Blanco Gaussiano, cte en todas *f*, nieve Big BangTV. Sol.: - Aumentar *Pot.* o Substr. Espectral - *AutoCorr.* concentrada.
 - Impulsivo o *Interfer.*: On/Off de *Dispos.* *EM* de gran *Pot.* como un motor. Sol.: - *Filtr.* Mediana (Sal y Pimienta) - *CRC*
 - *Diafon.* (Crosstalk): *Interf.* de naturaleza similar a la señal limpia (“habla cruzada”), normalmente por inducción *EM* de *Cable* similar cercano. Sol. 3 formas: - a) Conexión ok a *Tierra*. - b) *Par Trenz.* - c) *Apantall.* o envolviendo el *Cable*. - d) **IA Recon.**

Ver *LPC*, Cancelador R. (*RT DSP*)

- (*) Dispers.: le pasa a la luz de la *Fibra*
 - D. Intermodal (**por direc.** rayos) en F. Multimodo (MMF) los rayos *Luz* entran con distinto ángulo y unos tardan más que otros en llegar dando un *Puls.* ensanchado reduciendo el Ancho *Banda*
 - D. Cromática o *Refrac.* (**por frec.-vel.**, no confundir Intermodal anterior): la *Vel.* de los rayos *Luz* depende de la freq. y aunque sea luz monocromática nunca es perfecta y se nota la D. Indice R.: $n = \frac{c}{v}$ vel. Luz vacío vs medio. Ley R. Snell $n_1 \sin a_1 = n_2 \sin a_2$ (\in Ley Fermat (trayectoria de menor tiempo) \in Principio de Minima Acción *Fisic.* \in P. Variacional o extremal)

42.10 Otras

- (*) *Aten.* uación *Pot.* o Absorción (*Ley Beer-Lambert*): se debe la *Imped.* (*R* y *C*). Por *Shannon* a más dist. menos *Vel.* Ej: Fig. ?? *Puls.* de Fibra, también bordes se redondean por *Filtr.* armónicos.
 - *Metr.* en dB: $\alpha = 10 \log \frac{P_1}{P_2} = 20 \log \frac{V_1}{V_2} = 20 \log \frac{I_1}{I_2}$. Se pierde exponencialmente a la distancia recorrida y cuanto más denso sea el material. Ej: luz por líquido.
 - A. Intensidad: por Principio Conserv. *Energ.*, la I disminuye de forma natural como inv. prop. al cuadrado de dist (aunque la P es cte, ver ej. bombilla).

Ver *SNR* (*Canal Shannon*), *Medio Transm.*, *Ext. Red*, *Repet.*

⁶las f. fuera Canal son muy *Aten.* y no valen la pena transmitirlas

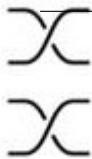
⁷poner la S. al rango OK del canal

⁸por *Shannon* disminuye A. Banda

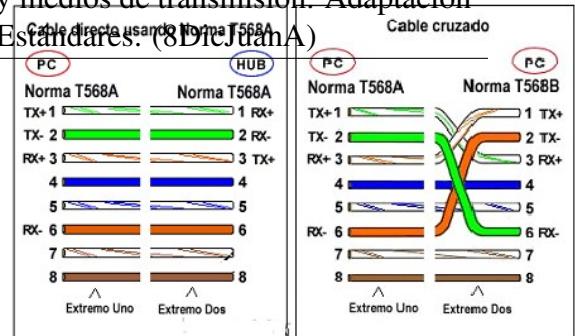
- (*) *Interfer.* o Conflicto (no confundir con *Interf.*): a) Signal constructiva o destructiva (*Cod. Land-Pit Disco CD*). b) Una cosa impide a otra su normal funcionamiento y ocurre por Compartir un *Recursos* o *Medio*, es *Unif.*. Ver *Multiproc.*, *Colis.*, *Sec. Crit.*, *Transac.*, *Par Trenz.*, *Coher.*, *Tierra*, *LIGO (Tele RT)*, Lamarr antijammin *Wifi*.

42.11 Estándarización de las redes

- *Estand. (Interop.)*, *ITU*, *ISO*, *IEEE*, *IETF (RFC)*
- *Serie 232*: reemplazado por lento por *USB* y *Ethernet*
- *ADSL*: 1 *Par Trenz.*?? reemplazándose por *HFC*, *Servic. Int.*
- *Móvil*: 3G UMTS, 4G LTE y 5G
- *MDI T568B* (Trenzado)
- *IEEE 802*: de las 4 *Ext. Red*: *Ethernet*, *Bluetooth*, *ZigBee*, *Wifi*,
- *ISO/IEC 11801 (Cable Estr. Par Trenz. y Fibra)*
- *ITU-T G.657 (SMF)*, *G.651.1 (MMF)* y *FDDI (Fibra)*



42.12 T63MedTransm



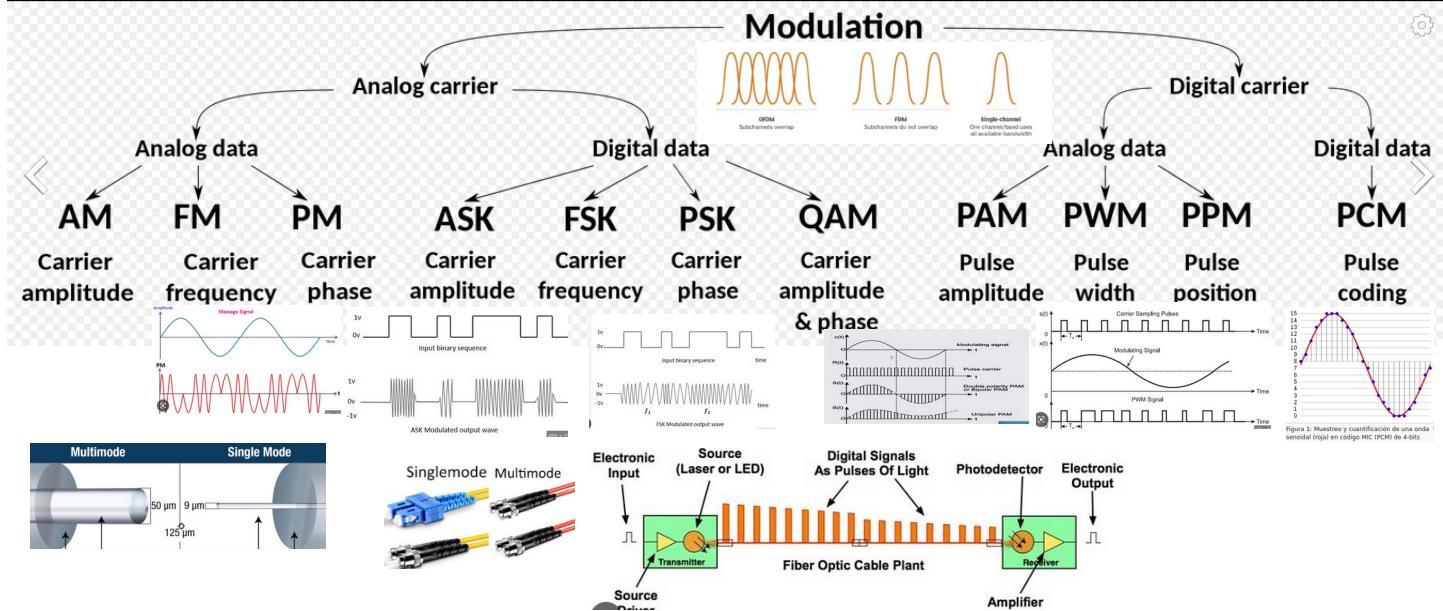
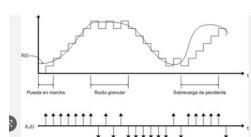
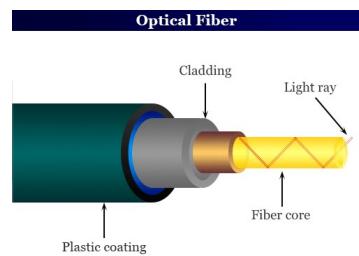
Category	Maximum Speed	Max. Length	Frequency	SHIELDING	Application
CAT 1	Up to 1Mbps(Carry only Voice)	--	1MHz	Unshielded	Old telephone cabling
CAT 2	Up to 4Mbps	--	4MHz	Unshielded	Token Ring Network
CAT 3	Up to 10Mbps	100m	16MHz	Unshielded	Token Ring & 10BASE-T Network
CAT 4	Up to 16Mbps	100m	20MHz	Unshielded	Token Ring Network
CAT 5	Up to 100Mbps	100m	100MHz	Unshielded	Ethernet, Fast ethernet and Token Ring
CAT 5e	Up to 1Gbps	100m	100MHz	Unshielded or Shielded	Ethernet, Fast ethernet & Gigabit ethernet
CAT 6	Up to 10Gbps	100m	250MHz	Unshielded or Shielded	Ethernet, Fast ethernet, Gigabit ethernet & 10G Ethernet(37 - 55 meter)
CAT 6a	Up to 10Gbps	100m	500MHz	Shielded	Ethernet, Fast ethernet, Gigabit ethernet & 10G Ethernet(37 - 55 meter)
CAT 7	Up to 10Gbps	100m	600MHz	Shielded	Ethernet, Fast ethernet, Gigabit ethernet & 10G Ethernet(100 meter)
CAT 8	Up to 40Gbps	100m	2000MHz	Shielded	Ethernet, Fast ethernet, Gigabit ethernet & 25G-40G Ethernet(30 meter)



Velocidad de transmisión de datos	Nivel de atenuación
4 Mbps	13 dB
10 Mbps	20 dB
16 Mbps	25 dB
100 Mbps	67 dB

Del CAT5 a 100m

Ancho de banda	100 KHz	1 MHz	20 MHz	100 MHz
En categoría 3	2 Km	500 m	100 m	no existe
En categoría 4	3 Km	600 m	150 m	no existe
En categoría 5	3 Km	700 m	160 m	100 m



Chapter 43

64. Funciones y servicios del nivel de enlace. Técnicas. Protocolos.

- **C. Enl.** : Err. (LLC) y Flujo (Colis.). ver OSI
 - Subdiv. del IEEE: MAC y LLC (def. la forma en que los datos son transf. por el Medio físico)
 - Ej: Ethernet, Wifi (IEEE 802) y ARP.
- **MAC** (Media Access Control, Control Acceso al Medio):
 - a) *Id.* o *Direc.* de Interlocutor de la NIC.
 - b) Control del orden de Comunic. de los interlocutores, lo hace la NIC. Ej: Colis. CSMA. Ej: Multiplex. (decir como hablan para no pisarse T/D/CDMA).
 - *Filtr.* MAC: de NIC para Segur..
 - **ARP** (Address Resolution Protoc.): en Ethernet para descubrir MACs dada la IP (ver ej. abajo). C. Enl. puede dirigir Paq. gracias a la Tabla ARP (MAC → IP). Es fundamental y fué definido en 1982 al comienzo del TCP/IP Hist. R.. En IPv6 Equiv. a NDP. El RARP (Reverse) tiene la Tabla Inv. y es anticuado y reemplazado por protocolo Bootstrap (**BOOTP**). Ver Subnet, Switch, NAT

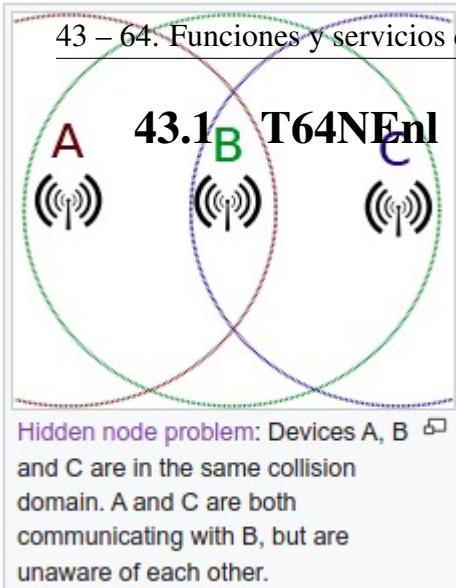
Ej. [ARP] Sea una LAN con 2 PCs conect. por un Switch.
0) PC Emisor consulta al DNS que le dice la IP del Receptor.
1) E. Busq. en su Tabla Cache ARP la MAC del R. (desde su IP).
2) Si no la encuentra emite un Broadcast MAC = FF : FF : FF : FF : FF a todos los PCs para que respondan con su MAC y E. Actual. su Tabla.
3) E. envia Trama a R. poniendo dicha MAC del R..

- **LLC** (Logic Link Control): mas a alto nivel que MAC. Lo hace la (NIC). Sobretodo Manej. de Err. y recepc. de Trama.
En Wifi el CSMA/CA Colis. hace el manejo y flujo por el LLC.
En Ethernet donde el poco probable los Err. no se hace ARQ (ver abajo, solo se detecta err. pero no se resolicita) excepto en caso Colis. que se hace CSMA/CD
 - **ARQ** (automatic repeat request, Petic.): usado en Wifi aunque creo no todos lo llevan. Si después de un timeout (TTL) no recibe AcRe el emisor retransmite (ACK y NACK). Similar al TCP. En Ethernet no se hace (ver LLC)

- *VLAN*.
- Protoc. *Arbol Recubr.* (**STP**, Spanning Tree P., no confundir con *Par Trenz.*): de *C. Enl.* que elimina *Circ.* o Bucles en *Ethernet* (y *VLAN???*) como *TTL* (Time To Live). Soluciona el *Probl.* Tormenta *Broadcast* causada por **Buckles Switch, Redund.** de caminos Físicos. Se permiten Bucles Fisico por *Redund.* (por si falla un Switch). STP permite los Bucles Fisic. pero elimina *Autom.* los Bucles Log. cuando se cambia un Nod. estableciendo un solo camino entre dos nodos.
El *IEEE 802.1aq* contiene (Shortest Path Bridging (SPB)) contiene al STP hoy.
 - Basado en Alg. de Radia Perlman (mujer): mientras trabajaba en *DEC*. Desactiva caminos que \notin Spanning Tree
 - *Alg. Kruskal*: *Alg. Voraz* crecemos el árbol (añadiendo nueva arista a_i pegada al árbol) evitando *Circ..* Ver *Jerarq. Broadcast*
 - STP (*VLAN*)
- *IEEE 802: Ethernet y Wifi. CSMA Colis.* y ej. de *Rend.*
- *Probl. Domin. Colis.* (no confundir con *C. Hash-SAM-Overflow*): *Interf.* entre *Tramas*, cuando en un Segm. dos *Nodo* trasnm. a la vez (aun esperando que hay baja prob. de que ocurra).
IMPORTANTE: Protoc. *MAC* de *C. Enl.* para evitar *C.* y hacer cierto manej. de *Err.* (ver *LLC*): **CSMA** (carrier-sense multiple access) que hay 2 tipos:
 - CSMA/CD (Col. Detection): ver si Libre para transm. era usado en los *Hub Repet.* *Ethernet* y en 2011 con la llegada de los *Switch* Ethernet quedaron obsoletos *Hist. R.*. Envía y si colis. hace pausa y vuelve a intentarlo (ver *Hub*)
 - CSMA/CA (Col. Avoidance): un Nod. antes de trasnm. dice que va a hacerlo para que los otros no lo hagan. El Probl. del *Nodo Ocult.* lo hace no fiable. Se mezcla con request-to-send/clear-to-send (RTS/CTS) handshaking para solucionarlo. Fig. 43.1. En los *Wireless* descentralizadas (*Moviles*) ocurre también ese problema. C. Típico de *Wifi* y *Ethernet* (antiguo con *Hub* ver, no en *Switch*).

Es *Unif.* por *Red Bus Compart. Recurso.* Ver *Hub, Congest., Monit., MD5 (Cript.)*.

Ej. *Ethernet* referente en *C. Enl.. Rend.* de una Red depend. del n° *Colis.*: $R(\%) = 1 - \frac{\text{Colis.}}{\text{Paq. Totales}}$



Chapter 44

65. Funciones y servicios del nivel de red y del nivel de transporte. Técnicas. Protocolos.

44.1 Nivel de Red

- Intro: Juanbe dice que hay bits de *Prior.-Confid.* de Info que envias: confid., secreto, hipersecreto, *Milit..*
- *C. Red* : encargada de Asignar IP y enrutar IP sin orientación a conexión. Es No *OrCo*. Ej: *IP*, ICMP *Ping*, ver *OSI*.
- *IP* : Dos significados:
 - *Protoc.* No *OrCo* y sin *AcRe* (*RFC 791*).
 - *Asign.* o *Direc.* IP: asignar de forma eficiente *Id.* a los dispositivos/nodos, i.e. físico (*MAC*)/lógico (*LLC* o *IP*) para crear redes y *Subnet*. Es el *Topos* de red. *Id.* o *Direc. Unic.* de Nodo. *Asign.* por la *ICANN-IETF*.

Ver *DHCP*, *NAT*, Screened Subnet (Des*Milit.* o *Firewall*), *IPSec* (*VPN*), *Popular*, *Gateway Home*

- *CIDR* (*Cl.assless Inter-Domin. Rut.ing*): método moderno de *Asign.* de *IP* que mejora EnRut. Lo estableció el *IETF* (IANA-*ICANN*) en 1993 para reemplazar al antiguo **Classful Network** de 1981 *Hist. R.* basado en:
 - Cl. A: desde 0.0.0.0 a 127.255.255.255 (/8 en notación CIDR)=muchos *Hosts* pocas *Redes*.
 - Cl. B: desde 128.0.0.0 (/16)
 - Cl. C: desde 192.0.0.0 (/24)
 - Cl. D: desde 224.0.0.0 (/4) para Multicast (*Broadcast*)
 - Cl. E: desde 249.0.0.0 a final 255.255.255.255 para invest. *Cient.*

Ver Red *Priv.*, *Subnet*

- Red *Priv.* o Dedicada (no confundir con InTranet o Red Propietaria (*Proveedor*)): Sus *Paq.* nunca son EnRut. a Internet (da *Vel.* y menos *Traf.*). Usa espacio de *IPs* privadas según el *RFC 1918 (CIDR)*.
 - Cl. A: 10.0.0.0 a 10.255.255.255
 - Cl. B: 172.16.0.0 a 172.31.255.255
 - Cl. C: 192.168.0.0 a 192.168.255.255

Ej: *Sucursal LAN, InTranet, IoT, VPN*. Lo contrario es una *PDN o Internet??*. Ver *Cl.*

- *NAT* (Network Address Translation, *Traduc.* de *Direc.* de Red): Op. de intercambio de *IP Public.* a Privadas y viceversa hecha por los *Rut.* (ver abajo). Esto permite al *Proveedor* dar *Internet*.
 - Lo hace el: *Rut., Gateway Home Priv.* o incluso *Maq. Virt.* (parar usar *NIC* del Anfitrión) en los *Paq.* de *I/O* (Fig.).
 - *IPv6*: elimina su necesidad. Además en *PC → NAT → Internet* en *IPv6* no es *Segur.*
 - Se hace en *C. Red con Masc..* Permite ahorrar *IPs*, *Manten.* y mejorar *Segur.* (Juanbe dice que no) pues hace invisibles al Exter. los *Hosts*. Similar al *Switch??*.
 - *FTP, RTP y SIP (Sesion Initiation Protocol)* son afectados por *NAT*.
 - *PAT: Puerto Address Translation*: en *DMZ (Milit.)* permite que con una *IP* externa se puedan responder hasta a 64000 direcciones internas.

Ver *ARP*

- *Masc.* : o *Map.* de bits. ver *Gateway, Ruido, Estr. Log. Fich., Asign. Esp. Libre (Sist. Arch., Trim o Gest. Mem.) Subnet. M. (VLMS) Interr. (Excl. Mut.)*.
- *Broadcast* (Difusión Amplia, no confundir con *Cast*): *Stream* de uno a todos.
 - En *IPv6*: Uni/Multi*Cast*...
 - En *Red Bus da Colis..* Ej: *IP, RF, TV, Satelite, Hub, tutor-padres-WhatsApp*.
 - Ej: Cl. D (*CIDR*)
 - *Domin.* de Difusión: conj. de Host que se alcanzan unos a otros mediante la Direc. B. de *C. Enl..* Usado en *VLAN*
 - *Probl. Tormenta B.:* ver *STP Recubr.*
 - *MiraCast*: usa *Wifi Direct* para *Screencast* de *Dispos.* a *Dispos* (*Stream* de *TV o Video*). Es *P2P* (o *E2E???*) y reemplaza a *HDMI*. Similar *Bluetooth* para *IoT*. Sopornado por *Nvidia* y *Android*)
 - *MultiCast*: en *Ethernet* las tramas con un valor de 1 en el bit menos significativo del primer octeto de la dirección de destino se tratan como multidifusión.

Ver *DAB y DVB-T (RF Digit., TV), Ethernet*

- *Subnet* (Subnetting o Subred): según la *Capa* usamos un *Dispos.* distinto. Reduce *Traf.* (va por *Grup.*) y mejora *Segur..* Con *Masc.* que *Part. IP* en *Red+Host*

–

- VLSM (Variable Length Subnet Mask, *Masc.* Subred de Tamaño *Var.*): *Subnet Variable*.
- *C. Física: Hubs*
- *C. Enl.: Switch* usando *MAC* (es *VLAN*)
- *C. Red:* usar parte de la direc. *IP* de *Hosts* para crear subredes. Ver *Cl. CIDR*
- *C. Apl.: Gateway*

S. *Variable* (VLSM, *Masc.*).

- ***IPv6*** : ver Problemas

- ya no hay *Broadcast* pero si *Uni/MultiCast*. ICMP (*Ping*).
- En *IPv6*: *UniCast* (de uno a uno), *MultiCast* (de uno a unos cuantos) Fig. 44.3.
- NDP (Neighbor Discovery Protocol): *Equiv. ARP*
- SLAAC-EUI64 $IP = f(MAC)$

Ver *NAT Segur*.

- ***TTL*** (Time To Live, no confundir con *TLS*): es un campo de 8bits en *Cabecera IP*, vale entre 255 y 0, se decrementa en 1 en cada *Rut.* y al llegar a 0 se descarta. Evita *Circ.* en *Rut.* ($R1 \rightarrow R2 \dots \rightarrow R1$). Como *STP Recubr.*. Lo lleva la cabecera de *IP* y *MPLS*. *ARQ* del *LLC* usa Timeout *TTL*. Ver *Plan*.
- ***Ping*** y ICMP (*Internet Control Message Protocol*): protoc. usado por Dispos. Red (PCs y *Rut.*) para *Testea* si \exists Conex. con otra *IP* (*OrCo*), es solo para comprobar, no para envia Paq. El *Comando Ping* o *Tracert=Traceroute* que *Testea Latencia* usa ICMP. Hay una versión ICMP para *IPv6* que auna varios anteriores como *ARP*. *IPv4* usa más **IGMP (IGP)**.
NOTA: para hacer Pings a PC de clase puede ser que tengamos que desactivar el *Firewall* Local de *Win Defender* (para que no bloquee el ICMP).

44.1.1 Protocolos de encaminamiento

- En *Rutamiento Efic.*: un *Paq.* según *Carg. Traf.*. Puede ser:
 - a) *Estat.* (*Actual.* de *Tabla* manual): “Paquete que va hacia: 192.168.3.0” “con máscara: 255.255.255.0” “siguiente salto (next hop): 192.168.2.2”
 - b) *Dinam.* (*Actual.* *Autom.* por *Protoc.* como *IGP*, mejor si *Fall.* un *Nod.* y por *Escal. Seis Grados*)
 Usa Mem. Asoc. para *Busq. Efic.*. *Filtr.* *Paq.* por *IP* (ver *Red Superp.*). *NAT* (ahorro de *IPs*).
- RESUMIR LO SIGUIENTE: ver hojas
 - *Probl. R. Más Corta:* es *P* (ver *Dijkstra*)!!
 - ***IGP*** (Interior *Gateway Protoc.*, no confundir con *Ping IGMP* abajo): *Protoc. Rut.. Ej.* de *IGP*: *OSPF* (de *Dijkstra* usado en *Frontera*), *RIP 1* (antiguos *Ruters*) o *2* (modernos), *VLSM* (*Subnet variable*).
 - ***MPLS***: trabaja con *IGP* y se basa en enviar no por *IP* si no por *Etiq.*. Ver *Seis Grados*

- EGP (Exteriores): BGO?? o BGP usado en R. *Frontera*
- IGMP (Internet Group Management Protocol): para *Video Streaming Broadcast* (Fig. 44.3). Ver *Ping*.
- *Sist. Autonom.*:
- *Tabla: Comando route (Ifconfig)*
- Alg. *Dijkstra (Turing)*: Alg. Voraz para *Probl. Rut.* más Corta en *Graf.*. Es *Opt.* en P y no debe confundirse con *Probl. Viajante que es NP*. **OSPF** (Protoc. *IGP* de *Rut.*).
 - Supongo *IGP* llegan por *Seis Grados*.
 - Bellman-Ford: ver Hojas

Otras sol. son Branch (*Back*) o Floyd-Warshall (ver *Parad. Progr.*). Ver Derivando. Ver *Estr. GoTo, Semaforo THEOS, InterBloq. (Filos. Chinos), Distrib. y Concurr.*

Seis Grados=Libre Escala (el grado de los nodos decae pot.)+Mundo Pequeño (saltos entre nodos crece log. con nº nodos)

44.2 Nivel de transporte

- *C. Transp.* : se encargada de enviar datagramas o Segmentos entre Emisor y Receptor final sin *Err.*. ver *TCP/IP* y *OSI*
 - Ej: *TCP (OrCo), UDP (NoOrCo)*
- *AcRe* (Acuse de Recibo, no confundir con *OrCo*): da Fiabilidad, el E. recibe OK o ACK (Acknowledgement en *ARQ* o *TCP*, cuyo carácter *ASCII* es 6₁₆) de confirm. de llegada *Integr.* (según *Checksum*) del Dato. Ej: *E2E, Pas. Mens., Sincr. ARQ (LLC)*, WhatsApp.
- *OrCo* (Orientado a Conexión, no confundir con *AcRe*): Implementado bien con *Comut. Circ.*.
 - 1º Establecimiento conex.:
 - 2º Comprobación: con *AcRe*.
El Recept. debe *Interr.* su tarea (*Cambio Ctx*) y esto le lleva un t., que puede aprovechar el Emisor puede hacer otra (IIO (*DMA*)), hasta que a) recibe *Signal de “Listo”* o b) con *Esper.* Activa ve que *Est. OK*.
 - 3º Uso
 - 4º Cierre
- Ej: *TCP, SMTP*. Pasos de *Comunic.* OrCo: Ej: *Ping, Socket (Perif., Arch. Dispos.), TLS*

No *OrCo*: Impl. en *Comut.* de Paq.. Cada Paq. (llamado *Datagram Segment.*) lleva la *Id.* de destino y cada uno se EnRut. de forma independiente. Ej: *UDP* Es *Unif.*

- Principio o *Filos*. **E2E** (End To End, Extremo a Extremo, Punto a Punto, no confundir con *P2P* ni con *PPP*): es la del *TCP/IP*. En *Dis.* Red y *Topol.* Red, es Red *Comut.* Paq. donde la responsabilidad de la EnCrip. (normalmente ASimetr.) y Fiabilidad de *AcRe* (*TCP*) reside en los *Nodo Terminal* (y no en las mitades que hacen *Comut.* de Paq. simple).

- Puede ser Simplex/Semi/Full/*Duplex*.
- Ej: *TCP*, *VPN*, *Serie 232*, *SATA* o *Walkitalki*, *TLS*

Es *Unif.*. Ver *PPP*, *MPLS*, *QoS*, *HyperTransport (Bus)*.

- *Puerto* (o *Conect.*): Nº *Id.* Log. de un *Servic.* normalmente en *C. Transp.*.

- *Firewall*: filtra los puertos
- 8100: *http://192.168.0.2:8100/* se conecta al Servid. Wifi-Proyecto del *Videojuego MineCraft* y *http://192.168.96.251:8100/* al Servid. LAN-Instituto
- 1433: *TCP Servid. BD* (ver *Gusano Malw.*) 80: *HTTP*, 443 *HTTPS*. 22: *SSH* 25 o 587: *SMTP* 20 *FTP*
- *Monit.* o Escaneo P.: ver *Malw. Gusano*.
- Otros: P. *Micro*.

Ver *PAT (NAT)*, *PMIO*, *DMZ (Milit.)*, *I/O*, *Socket*, *Switch vs Brigde*.

- *Socket* (Enchufe): es un tipo de *Obj. API* normalmente para Red (aunque también *Driver* de cualquier *Perif.*, *Web (Scraping)*, *Audio*, *USB*, ...). Pasos de *Comunic. OrCo*: 1º establecer conex. 2º usarla, 3º cierrarla. Es un Obj. donde definimos la IP/Puertos local y remota, el Protoc. *C. Transp. (TCP)*. Ej: S. *TCP* en *C*

```
s = socket(AF_INET,SOCK_STREAM,0); // Create Socket
if( bind(s,(struct sockaddr*)&client , sizeof(client)) < 0) { perror("bind");} //Connect Client to socket
connect(s,(struct sockaddr*)&server , sizeof(server)); //connection
close(s); //close socket
```

Ej: S. *Robot (Arch. Dispos.)*, es un *Descr. Arch. (INode)*

.... ponerlo

Es un *Unif.*. Ver *Zócalo (CPU)*, *Bibl. BGI (C)*.

Ej. *Socket*

- *TCP* (Transmission Control Protocol): *OrCo* con *AcRe* (ACK) (*ARQ*), Control *Err.* y Control *Flujo* Datos, los *Segment.* llegan en orden, es un *Circ. Virt* (ver abajo).

- *Circ. Virt.* (VC, no confundir con *Red Superp.*): tipo de Red de *Conmut.* Paq. para una comunic. *OrCo* en la que los datos del Emisor al Receptor pueden viajar por más de 1 circ. pero que es *Transpar.* para ambos. Ej: clásico *TCP*. Otros ej.: *Frame Relay*, X.25.
- *MSS* (Maximum Segment. Size): 536Bytes ver ej. *Ud.*

Ver *Puertos*

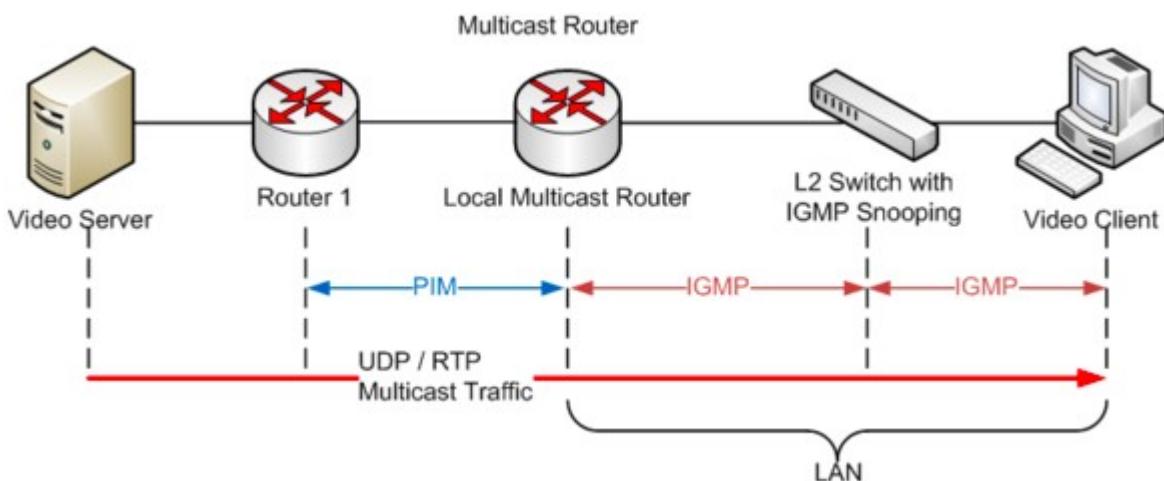
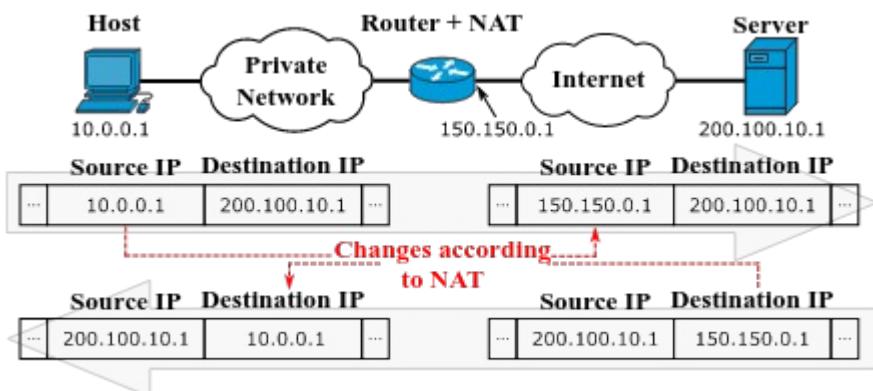
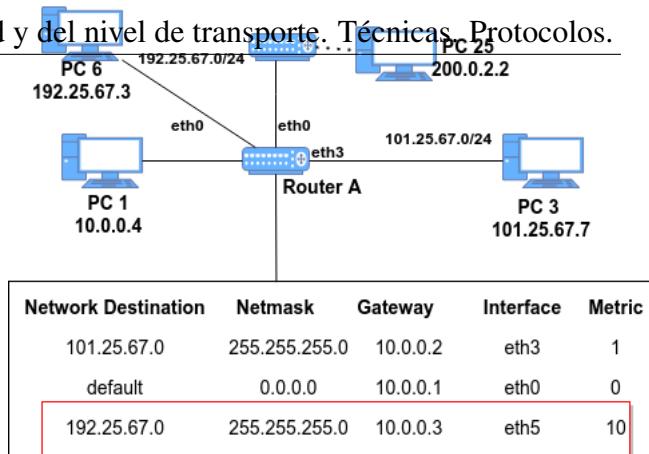
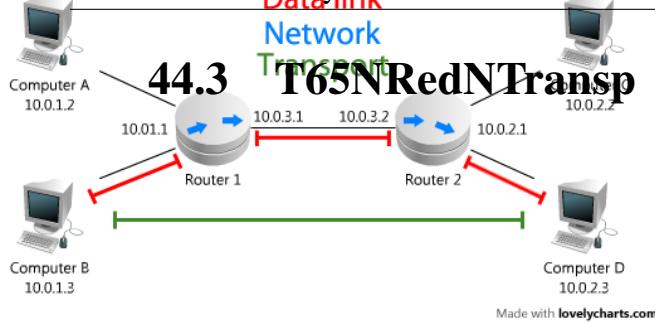
MTU (Maximum Transfer *Ud.*): al establecer *TCP* los 2 Comput. deben acordarlo y suele ser 576Bytes. *MSS* (Maximum Segment. Size): luego *MSS* = *MTU* – *CabezaTCP* – *CabezaIP* tamaño *Block Payload*, y por defecto es 536Bytes. Tanto en *TCP* como en *IP* su *Cabecera* es de 20Bytes.

- *UDP* (User Datagram (*Segment.*) Protocol): No *OrCo* y sin *AcRe*, sin Control *Err.* y sin Control *Flujo* Datos. Ver *SNMP Monit.*, Ej. *Fork. Stream* con *RTP*

- **RTP** (*RT Transport Protoc.*): la *ETSI* creó el *RFC 3557* y el *IETF* el 3550 en 2003. Basado en Emisor *Emisor*. Paq. *UDP+Receptor* *Ord.* en *Buffer* (Fig. 29.13 *Cabecera*). Esto parchea el que *Internet* no es *RT*. Cumple Princ. *E2E*. Sirve para adaptar *QoS*. Otros para *Streaming* (*TV o VoIP o Multimedia*) son:

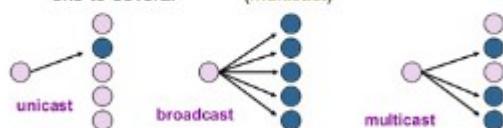
- *RTCP*: usar RTP para
- *WebRTC*: *APIs* para RT en Web
- *RTCP*: de C. *Sesion*

Ver *IoT*.



IP Service

- IP supports the following services:
 - one-to-one (unicast)
 - one-to-all (broadcast)
 - one-to-several (multicast)



- IP multicast also supports a many-to-many service.
- IP multicast requires support of other protocols (IGMP, multicast routing).

Chapter 45

66. Funciones y servicios en niveles sesión, presentación y aplicación. Protocolos. Estándares.

45.1 Sesión

- *Sesion* : ver *OSI*.

- Ej:

Ver *Rest., Autent., Admin.*)

45.2 Presentación

- C. *Present.* : ver *OSI*. *Serielizar*.

- Ej: *JSON*.
 - Suele ser amigable y *Compr.esible* al usr.

Ver *Binar. Cod., Arq. MultiTier, SGBD*

- *Serielizar* o marshalling: *Lin.* o *Secuenc.*, *Form.* para *Transm.* en Red una *Estr.*, permite la *Persist.*.
- *JSON* (*JavaScript Object Notation*): *Estr. Dat.* no *NoSQL* y *Serielizada*. Es de C. *Present.*. Para intercambio de datos entre cliente y *Servid.* usando (como *Struct*). Compite con *XML* o *ASN.1*. Ver Jakarta (*Progr. Compo.*). *Kubernetes*.
- *XML* : *Estr. Dat.* *Serielizada* (ver *JSON*) y *NoSQL* que compite con *JSON* y está basado en SGML como *HTML*. Se está usando en *Robot* con *Apl. Web*: <https://niryo.com/2017/03/robotics-web-applications/>. Es de C. *Present.*
 - VoiceXML: conj. de Tags (*Etiq.*) que ejecuta un Voice Browser para ASR y Sintesis de Voz. Del *W3C*
 - *Music. XML*:

- *Model. BD Jerarq.*: usa XML, para geografía (ver)

Ver *SGBD DB2. ASN.1. Jakarta (Progr. Compo.), Struct, XQuery (SQL)*

- *ASN.1* : *Protoc.* para *rePresent.* datos independ. de la máquina. Especif. por X.690. Para *Telecom.*, la *Cript.* y la biometría. Reemplazador por *JSON* y *XML*.

45.3 Aplicación

- *C. Apl.* : ver *TCP/IP OSI*
 - Ej: *SNMP* (para *Monit.* y *Admin.*).
- *HTTP* : *RFC 2616* escrito por Tim Berners-Lee (*Hist. R.*). Transferir *Fich.* de Hypertext como *XHTML (XML)*, *HTML* (sirve como *Servid.* de Arch). Este *Protoc.* ha creado la *WWWeb* (regulada por la *W3C*). *Puerto 80* y *443 (HTTPS)*. Ver *Cookie*
- Word Wid *Web* (WWW): creadas para mostrar *Multimedia* (en cualquier *Dispos.*) y *Vincul.* (*Sindic.*, *Cache*). Podemos *Embeb.* otras W. o *Multimedia (Recurs.)*. *Texto es Imag. Vect.*. Por Lee en CERN Turing A.
 - Web2.0: Interactiva como *TV Digit.* o *SmartTV*. con *JavaScript*.
 - Web Responsive o Adaptativa: se *Visual* bien en cualquier *Dispos.* (*Movil, Tablet..*) *Win, Ubuntu Server..*
 - URL (Uniform Resource Locator, *Recurso*): *Direc.* para *GET* en *PHP*. Ej. *http://www.ugr.es* usa protocolo *HTTP* y está un *Servid.* con Direcc. *www.ugr.es*. Acortar (ver *Phishing Suplant.*)

Ver *HTTP, HTTPS y Servid. Web, Apache, SSH, W3C, Unicode, WebUSB (IoT), W. Semant. (HTML5)*

- *Frontend* (no confundir con *Frontera*) y *Backend* (ver *Node.js JavaScript*): *Arq. MultiTier* referente de la *Web*. Es *Present.* vs *Capa Acces.* a *BD (Persist.)*. Ver *MultiTier (SGBD)*.
- *FTP* (File Transfer Protocol): es el más *Popular* basado en *RFC 959*. *FTPS*(mejor nombrado como *FTP/SSL* y basado en *SSL/TLS*) no debe confundirse con *SFTP* (mejor nombrado como *SSHFTP* ver). *Filezilla* es una Apl. Bueno para Transf. Arch. *Masiv.*. *Puerto 20*. F. Ver *Samba*.
- *Email (SMTP* (*SMTP* es *OrCo*), *POP3, IMAP*). Client. (Apl.) de E. *Popular*: *Puerto 25* antiguo y *587* el moderno.
 - *Libre*: *Nginx (Servid. Web de E.)*, *Thunderbird (Client. de E. de Mozilla)*
 - *Servid. E.: MDaemon de Win* (soporta *IMAP, SMTP,..*) y *MSExchange Win (Client. Servid. de E.)*, *MS Outlook (Client E. pero también calendarios compartidos..)*, *Evolution (Linux, email, calendario.. para trabajo Cooper.)*
 - *Filtr.* de E.: *Spam*
 - *PGP (Pretty Good Privacy)*: con *Cript.* Hibrida (*Simetr. y AES*)
- *DHCP* y *DNS*

Chapter 46

*+67. Redes de área local. Componentes. Topologías. Estándares. Protocolos. (3DicMariCarmen)

46.1 Introducción. Conclusión

- Ej-Motivador: ¿Como hacer un *Almac. Distrib.* o *TAP* cercano.? Sol. *Cluster* usa LAN.
Ej: ¿Qué diferencia hay entre un *Switch* y un *Rut*.
- Hist.: Ver Fig. 46.13 Serie 232??,
 - a) *IPX/SPX* perdió frente a *TCP/IP* en LAN.
 - b) *Token Ring* y *FDDI* frente a *Ethernet*.
 - c) [Boggs,Metcalf] *Ethernet*
 - d) [Lamarr] *Wifi*.
- Futuro: *USB-c+Ethernet*, *MiraCast*, *Ethernet Fibra* y *Repet. Luz. IoT Par Trenz.* y *Wifi* más *Vel.* y menos *Aten.*

46.2 Arquitectura de redes

- *Arq. Red, Pila Protoc., Tab. TCP/IP vs OSI* (Amb. Influencia), *Encapsul. de Paq. (PDUD.)*

46.3 Red de Área Local

- **LAN** (Local Area Network): *Red* del *Ext. Red* de una *Home* o *PYME* (Pequeña Y Mediana Empresa). Para *Comunic.* y *Compart.* *4 Recursos (Servic. Dir.)*. Las 2 *Tecn.* más comunes son:
 - *Ethernet* y *Wifi*.
 - *IPX/SPX*, etc.: se usaron en LAN pero ganó *TCP/IP*. *Hist. R.*

Ver Red *Priv.*, *WakeOnLAN (Rest.)*. *SAN (Almac. Distrib.)*

- Características de LAN:

- *Vel.* Transm.,
- *Ext. Red,*
- *Topol.* Red y *Protoc.*
- Tasa *Err.*
- *Precio*
- Compart. *Recursos*

46.4 Topología: red bus

- **Topol.** Red: *Def.* de la *C. Fisica* de como se conectan los *Nodo*. Permite *Acces. Transpar.* a cualquier Nod. Alcanzable (*Ambito*) independ. de su localización geométrica (*Topol.=1*a ventaja de *Red, Telepresencia*). Elegirla depende de multiples factores (nº maquinas, espacio disponible, tipo *MAC..*). Nombremos las T. Físicas, luego las T. Lógicas. Ver Packet *Tracer CISCO, Plan.*
- **Red Bus** : es la esencia de un *Hipergraf.* donde todos los *Nodo* conect. al mismo *Canal (Cable Backbone)*, entonces da *Colis.* (y llevan CSMA).
Ej: *Hub, Movil, Wifi... .* Ventaja: *Robust.* (no cae si Nod. *Fall.*), fácil, barata. Desventaja: *Colis..* (difícil *Manej. Err.*). Es *Unif.* (compartir *Recurso*). Ver *Token Ring, Bus, Multiplex., Broadcast, UniBus (MIO)*
- Estrella (Star): Todos los N. conect. a centro (normalmente Dispos. *Conect. WAP*). V: Robust., facil Err. D: No Robust. si centro Falla.
Ej: **Client.-Servid. (Compart. Recurso)**, Top. Red Infrastr, la más *Popular* (casa, router, clase)
- Anillo (**Ring**): cada uno con al lado. **Doble Anillo** (*Redund.* para evitar Caidas).
Ej: en *Fibra*. Ver *Token Ring*. V: bajo Err. (señal *Repet.*). D: no Robust si 1 falla.
- **Arbol (Tree)**: *Graf.* con min. nº de Arcos luego V: barato (*Redund.* min. en *Canales*). D: Poco *Robust..* Poco usada. Ver *Recubr.*
- Malla (Mesh): *Graf.* totalmente conexo pero donde falta alguno. V. y D. Contrario a *Arbol* muy Robust pero Caro.
Ej: **P2P** (SO ligeros *Compart. Recursos* en igualdad de condic.)
Ej: (Maq. Boltmann de Hinton *NN*)
Ej: *Cluster*: Red Proc. con SO *Distrib.*
- Otros: *E2E, Mixta, Fract. o Seis Grados* (Libre Escala y Mundo Pequeño)
- **Topol.** Red Lógica (o por *Softw.??*): forma de transm. Tramas entre *Nodo*. Más de *Wireless Wifi*
 - Ad-hoc: *Comunic.* entre Pares (*E2E*), solo requiere *Tarj. Red.* No requiere *WAP*.
Ej: *PAN Bluetooth??* o con *Cable*
 - Infraestructura: *Topol.* Red Estrella con *WAP Central*.
 - Malla (Mesh): Mixta, algunos nodos comunicados entre si y otros con *WAP*.
Ej: Red Domestica, *Wifi City* (Ciudad Inalambrica, donde los ciudadanos se conectan a distintos servicios municipales)

46.5 Componentes: clasificación

- *Hardw. Red*: ETD, *Medio*, ECD. Tabla para situar *Switch*.

46.5.1 Componentes: Switch

- *Switch* (o *Commut.*): en *C. Enl.* conecta *Hosts* dentro de una misma *Red* (*IP* fija terminada en 0) mediante *Commut. Paq..*
 - *Filtr.* hardw. las *MAC* (descubiertas con *ARP*). Esto es más *Segur.* y aprovecha mejor el A. *Banda* (del que además decide *Equil. Carg., QoS*) que el *Hub*.
 - *VLAN*: (abajo). Algunos S. también pueden añadir funcionalidades del *Rut.*, filtr. en la *C. Red* llamandose S. *MultiCapa* o *Layer-3* o *Modelo Gestionado*.
 - Tienen un *NOS* (*Comandos Ifconfig*), un *Patch* y puede tener *MDI* (como mi *TP-Link*). Ej: S. de Aula Informática. Similar al *Gateway*??
 - *Brigde* : en *C. Enl.* conecta solo 2 (o 3) segmentos *LAN* con mismo protocolo (como *Ethernet*) y *Filtr.* por Softw. analizando *MAC* de *Tramas*. Ver *Chip Puente Norte*.
 - B. vs S.: S. es un B. *multiPuerto* (permite crear *Subnet VLAN*) y toma mejores decisiones que B.. Tiene *Buffer*, más interfaces (≈ 30) y filtra por Hardw. (mientras que B. es *UniPuerto* simple, no *Buffer*).
- *VLAN*: por *Switch* de L3 (ver), permite al *Admin.* crear *Grup. Config.* el *Switch*. Puede Transgredir IP.

46.5.2 Componentes: gateway residencial (modem), WAP y tarjeta

- *Home Gateway*: converg. *Tecn.*
- *WAP*: *Vuln.* *WPS*
- *NIC*: $PC(DriverEnSO) \rightarrow Tarj.Red(Firmw.) \rightarrow Red$

46.5.3 Componentes de instalación: par trenzado y racks

- *Par Trenz.*: Cat5 (ver abajo *Fast/Giga Ethernet*)
- *RJ45*
- Canaletas *Cable*
- *Herram. Instal.*: fusionadora vs crimpadora, tester, puncher, rosetas...
- *Rack, Patch,*
- *PLC Red* y *HyFi*

46.6 Componentes Software

- *Softw. Red: Arq.* referentes Client.-Servid. y Pila Protocol.
- Para *LAN e InTranet*: SO Distrib. (*Cluster*), *SO Servid.* (Active Dir.) vs Client. *Protoc.* y *NOS, Monit.* Red...

46.7 Estándares: IEEE 802

- *IEEE 802* : Estand. IEEE de Facto para *LAN* y también *PAN o MAN* para capas inf. del *OSI* (*C. Fisica y C. Enl.* de los 3 *Hardw. Red Hub*,..).

	Descrip.	<i>Medio, Ext. Red Vel. Transm.</i>	Otros <i>Err.</i>
<i>Ethernet</i> (802.3)	LAN <i>Cableada</i>	500m/1Gbs (Trenz.), 10km/100Gbs (Fibra)	Cod. <i>Manchester</i>
<i>Wifi</i> (802.11)	LAN <i>Wireless</i>	100m/2Gbs	<i>Colis.</i> CSMA/CA, <i>MiraCast</i> para <i>Video</i>
<i>Bluetooth</i> (802.15)	WPAN	10m 1Mbps	BLE (<i>IoT</i>)
Otros: <i>ZigBee</i> (802.15.4), <i>Token Bus/Ring</i> (802.4/5), <i>WiMAX</i> (802.16), <i>VLAN</i> (802.1Q) <i>STP Recubr.</i> (802.1aq)			

- *Bluetooth* (IEEE 802.15): para WPAN Voz y datos lentos:
 - 10m 1Mbps
 - en RF 2.4GHz y puede *Colis.* *Wifi* 802.11b. Tiene *CRC*), BLE (*IoT*)

46.8 Ethernet

- *Ethernet* (IEEE 802.3) (.3ab??): Para *LAN Cableadas* (*Coaxial, Par Trenz.* o *Fibra*) y distingue:
 - Ethernet (original): 10Mbps (Coaxial)
 - Fast E. (antiguo): 100Mbps Ej: 100BaseT (100m, Par-Cat5) o 100BaseLX (10km, Fibra-mono.)
 - Giga E. (hoy): 1Gbps Ej: 1000BASE-T (100m, Par-Cat5) o 1000BaseLX (10km, Fibra-mono.)
 - 10 o 100 Giga E.: 10 o 100 Gbps Ej: 100GBASE-SR10 (100m, Fibra), 100GBASE-ZR (80km, Fibra)
 - Model. *Jerarq. Red:* de 3 Capas de *CISCO*, se basa en E.

- *C. Fisica:* Cod. 64b/66b y NRZ 100 Gigabit Ethernet. Tiene *CRC* por C. Serie. Antiguamente C. Lin. Manchester.
- $MTU < 1500B$ (*Ud.*) sino Jumbo Frame (ver ej. abajo)
- *C. Enl.:*
 - Control *Flujo* Datos [Ethernet flow control] que para envio o *Prior.* (ej. VoIP sobre IP) en caso de *Congest.* para evitar *Perd.* dat.
 - Evita *Colis.* (en antiguos *Hub* con CSMA/CD) y *Circ.* (con STP *Recubr.*).
 - Trabaja con *ARP* para descubrir *MAC* de *IP*.
- ETTH: en la “Last Mile” lleva *Internet* al *Home* (como *Fibra FTTH*). o *PPP* obre E.
 - (SPE): Single Pair Ethernet en vez de 4 solo 2 cables. (pesa menos).
- Boggs, Metcalfe 73 Premio *Turing* Hist. R. Nombre del Eter=Ether. E. WAN (*ATM*). Reemplazó a Token Ring y FDDI (*Fibra*)

Ver *Broadcast*, *VLAN*, *HyperTransport*

En *Ethernet* la *Trama Max* es $MTU = 1500Bytes$ (*Ud.*) para hacerla mayor usar **Protoc. Jumbo Frame**

46.9 Wifi

- **Wifi** (Wireless Fidelity IEEE 802.11): Inalambrica para *LAN*, edificios y plazas. Ej: Wifi City (ver *Topol.* Red Mesh).
 - *ITU Microondas* corto alcance <100m según Ley:
Wifi5'14 (0,6Gbps, 5 GHz, 802.11b),
Wifi6'19 (9Gbps 2'4/5 Ghz, 802.11ax),
Wifi7'24 (46Gps, 2'4/5/6 GHz)
 - **SSID** (service set *Id.*, nombre puede ser Ocult. *Encapsul.*, *CISCO Config.*).
 - *Cript.* *WPA* (*Intercep.*)
 - *CSMA/CA* (*MAC*) para evitar *Colis.* (solo emitir si canal inactivo, *Red Bus*). y *ARQ* (*LLC*)
 - Funciona con varias *Topol.:* *WAP*, Red Ad-hoc (sin *WAP*). Mesh (Topol. exteriores Wifi6??).
 - Botón **WPS**: ver *Vulner.* (*WPA*).
 - Wifi se come al *Cable*: Ej: MiraCast (ScreenCast=Stream Video) vs *HDMI*, *M.2*: para Conect.
 - Hedy Lamarr: actriz austriaca, creo en 1942 el **espectro ensanchado** (un caso de *Cript.* y *Modul.*) que llevan hoy. Fig. 46.13. La propiedad antijamming (capacidad para evitar las *Interf.* intencionadas, se basa en formula *Shannon*.)

Ver *Lifi*, *WiMAX*

46.10 Colisiones: VLAN y circuitos en Ethernet

- *VLAN* (Transgresión IP) y STP *Recubr.* ayudan a reducir *Colis.* especialmente en *Wifi*. CSMA/CD vs /CA,

Fórmula del *Rend.* en *Colis.*

46.11 Protocolos de acceso a Internet

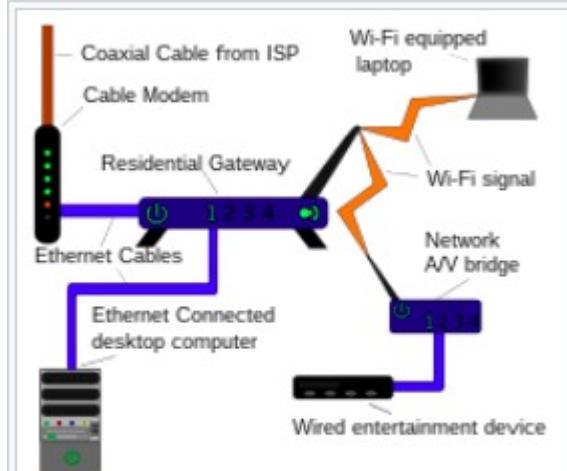
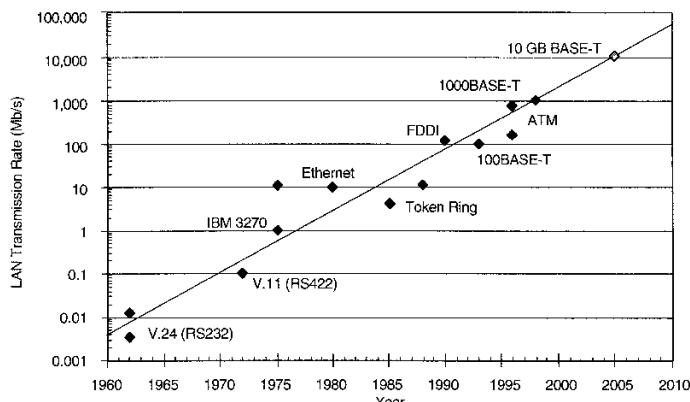
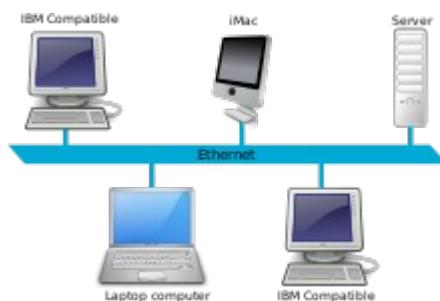
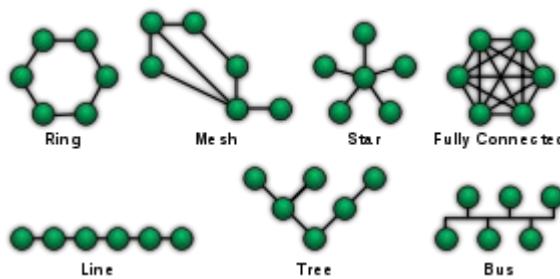
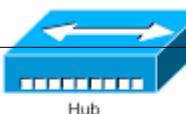
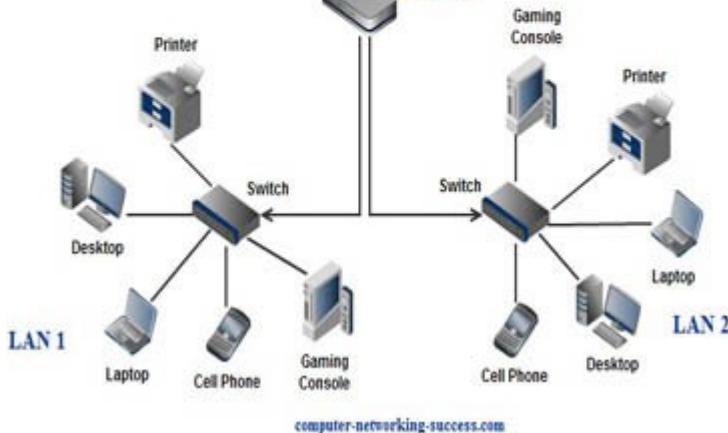
- En LAN las 2 o 3 Capas *OSI* más bajas son las importantes.
- Para Acces. a *Internet* (ver):
 - *PPP*: conect. 2 *Rut.* para acceder a *Internet* mediante *ADSL* con *Servic. Int.* RDSI.
Ej: *PPP over Ethernet*
 - *ADSL*
 - *NAT*
 - *ARP*: para descubrir *MACs* ver ej.

Ej. *ARP*

- *VPN*: permite usar Red *Public.* para comunicar 2 LANs.

46.12 Protocolos antiguos

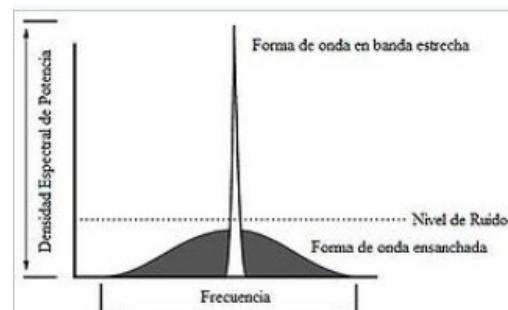
- *Token Ring* o *T. Bus* (según *Topol.*): *IEEE 802.4* o *5* usado en FDDI doble anillo (y **ahora en Fibra Tier2, Juanbe, Marcos y MariCarmen**). Elimina el problema de las *Colis..* De *IBM70* substituida por *Ethernet Hist. R.*
 - Usa *Cod. Manchester* Diferencial.
 - **FDDI** (*Fibra Distrib.uted Data Interf.*): protocolo de F. para *LAN* pensado para sustituir *Ethernet* pero no lo consiguió y en 2000 le ganó Gigabit *Ethernet* al ser más barato y rápido *Hist. R.*. Con Comunic. *Duplex* y basado en *Token Ring*. Se hizo uno similar para Cobre CDDI. Usada como *Cable Backbone WAN* (ver).

Local Area Network**46.13 T67LANCompProt**

Esquema de un ejemplo de red doméstica sencilla.

Ethernet Name	Cable Type	Maximum Speed	Maximum Transmission Distance	Cable Name
100Base-TX	UTP	100Mbps	100 Meters	CAT5, CAT5e, CAT6
1000Base-T	UTP	1000Mbps	100 Meters	CAT5e, CAT6
1000Base-SX	Fiber	1000Mbps	550 Meters	Multimode and Singlemode Fiber
1000Base-LX	Fiber	1000Mbps	550 Mbps MMF, 2000 Meters SMF	Singlemode Fiber
1000Base-ZX	Fiber	1000Mbps	70000 Meters (70 Kilometers)	Singlemode Fiber
10GBase-T	UTP	10Gbps	100 Meters	CAT5e, CAT6
10GBase-SR	Fiber	10Gbps	300 Meters	Multimode Fiber
10GBase-LR	Fiber	10Gbps	10000 Meters (10 Kilometers)	Singlemode Fiber
10GBase-ER	Fiber	10Gbps	40000 Meters (40 Kilometers)	Singlemode Fiber
10GBase-SW	Fiber	10Gbps	300 Meters	Multimode Fiber
10GBase-LW	Fiber	10Gbps	10000 Meters (10 Kilometers)	Singlemode Fiber
10GBase-EW	Fiber	10Gbps	40000 Meters (40 Kilometers)	Singlemode Fiber

Multimode Fiber Singlemode Fiber 10G Multimode Fiber SFP+Copper (Twisted Pair)



Comparación de una señal en banda estrecha con una señal modulada en **secuencia directa**. La señal en banda estrecha se suprime al transmitir el espectro ensanchado.

Chapter 47

+68. Software de sistemas en red. Componentes. Funciones. Estructura. (4FebLola)

47.1 Introducción. Conclusión

- Ej motiv.: imagina empresa con muchos PCs y hay que *Actual.* ¿*Pila Protoc.* ventajas? permite implementar en Switch solo 1MB de los 10MB ¿*Monit.* red?
- Hist.: *Netware* 1er Active Dir. de los 1º *SO Servidor* que permitia compartir *Recurso* en red.
Win NT 3.1 Adanveced Server 1993, luego *Win Server* 2022
- Futuro: *Kubernetes*

47.2 Sistemas en red

- *Red:* 1º ventaja *Tele.*
- *Hardw. Red vs Softw. Red:* ETD (*Servid.*, Impresoras), *Medio Transm.* (*Fibra*)
- *Estand.:* ISO, IETF

47.3 Software de red. Clasificación. Importancia de los SO

- *Softw. Red*: parte funcional (*Softw.*) de *Arq. Red* para explotar y *Admin.* la Red.
Podemos distinguir según: a) *Nodo Terminal* (ETD) o de *Comunic* (ECD) como en *Hardw. Red*. b) *Nivel OSI* Nosotros los separamos en:
 - A: *Driver Impr.*
 - SO: *NOS* (abajo). Es el principal Compo. y la Arq. Referentes son Client-Servid.

- BD: *PhpMyAdmin*, *Servid. BD*:
- L.: *C, HTML, PHP, JavaScript*
- R.: ver TAP y *Integr.*: *Mashup, Scraping*
- *TAP*:
T: *Pila Protoc. y Estand.*:
A: *Almac. Distrib.* ver abajo
P: *Cluster PBS qsub, Apl. Web*:
- *NOS, SO Servid. (Netware, BDS, Solaris, Stack Soluc.)*.
- *Almac. Distrib. NFS Samba*
- *Servic. Dir. (Active Dir.), LDAP, Remot., Veyon, SSH*
- *Monit.* y auditorias: *SNMP, Pandora FMS, logcheck (Ubuntu)*.
Control: Kubernetes
- *Simul.*: *Packet Tracer*

47.4 Softw. red: funciones principales

- *Monit.*
- Servic. *Email* (ej. ugr), Comunicar *LANs* entre si, Polit. de *Segur.* (en Dat. y *Autent.* sin *Suplant.*)
- *DNS* (no memorizar IP), *DHCP* (asignar parámetros de red), *Servic. Dir. (Active Dir.)*
- Terminales (*SSH*), *SFTP* (Filezilla)
- *Servid. BD (MySQL)*, *Servid. Email (SMTP Nginx)*, *Servid. Web (Apache)*

47.5 Sistemas operativos de red

- ***NOS*** (Network OS): *SO* principalmente de *Nodo Comunic.* como *Rut.* y *Switch* como el ***CISCO IOS*** (con *CLI*). Para *Config. Puerto*, *enRut.*, *VLAN...*
Aunque yo los distingo, también incluye los *SO Servid.* y los de *Firewall*. Tiene *Comandos* (ver *Ifconfig*).
 - *SO Servid.* o Pesados: *Servic. Dir.* con *Obj. (Usr, Fich...)*
 - *SO Ligeros*: los *Acces.* a Recursos son *Transpar.* al *Usr*. Se puede convertir en un *SO Servid.* con *Stack Soluc.* (ver). Los *Populares* son *Win11* y *Ubuntu22, Android*. Para *P2P*
 - *SO Distrib.* de *Grid* o *Cluster*

47.6 SO ligeros: Stack solucion

- **Stack Soluc.** : Pila o conj. de Apl. que instalarlas por separado cuesta pero que juntas permiten tener una plataforma como *Servid. Web*. Ej: XAMPP (cross-platform in OS) con *Apache*, *MariaDB* or *MySQL* (database con *PhpMyAdmin*), *PHP* (programming language). Perl (programming language)). Ej: LAMP con *Linux*, *Apache*, *MySQL* y las 3 Ps (*Perl*, *PHP*, *Python*). Ej: WIMP: con *Win*, Internet Information Server (IIS) (de *Win* convierte a PC en *Servid. Web*) y las 3 Ps. Ver *Daemon*,
- **Mashup** : una *Apl. Web* llama a otras con (*APIs* o *Scraping*) y te muestra el resultado final. Requiere *Interop.*.. Es un *Unif.*.. Ver *Sindic.*, *Scraping*
- **Scraping** (Raspado Web): con *Selenium* y *Socket*. Ej: a) resumir en web ofertas de trabajo de muchas web. b) llenar *Formularios*
- *HTML, Php, JavaScript*

47.7 Sistemas operativos para servidores: estructura

- **SO Servid.** (SOS ⊂ NOS, no confundir con SO Distrib. Cluster): SO pesados para *Servid.* y *Admin.* muchos (*Masiv.*) *Client.* (con SO Ligeros) o los *Obj.* (ver *Servic. Dir.*) de una red. Distinguimos:
 - **Servid. No Dedicado:** *Client.* que cede parte de sus Recursos para ser Servidor. Es un cualquier PC con SO Ligero con un *Stack Soluc.* como ISS o XAMPP. S. **Dedicado** (más RAM, RAID, varias conexiones de red....).
 - * **SO en Red** (Módulo de 2º SMR): en una empresa hay un ordenador *Admin.* donde se instalan las Apl. para Usr y se guardan los datos de Usr. Gestiona 4 *Recursos Red* (*Servic. Dir.*)

Ver *Softw. Red*

- *Win Server*: SOS para:
 - a) *Admin.* muchos *Client.* (*Recursos u Obj.*) mediante el Active Dir. (*Servic. Dir.*), b) asignarles *IP* mediante un *DHCP Server*, c) agruparlos/localizarlos en dominios mediante un *DNS Server* d) darles *Segur.:* con *Autent.*, *Actual.* (*Win Defender*) o creando de *Arboles Group Policy*. e) Otros: *Servic. Escrit. Remot.* f) Otros???: *Servid. Web, Proxy, FTP, VPN*
 - *Hist. Win:* el 1º en 1993 (*Win NT (New Technology)* 3.1 Advanced Server), desde 2003 se llaman *Win Server*, los últimos son 2019 y 2022. Tipos: Essential, Standard y Datacenter.
- *Ubuntu Server*: SOS. *root* es *Admin..* Solo en modo *CLI* pero se puede hacer *GUI* añadiendo (X Window Environment con GNOME, KDE o Xfe). Es certificado para *Amazon AWS* o *Azure (Nube)*. Fácil *Backup*. Se pueden instalar paquetes como:
 - *OpenLDAP: Servic. Dir.*
 - *Stack Soluc.:* como LAMP o XAMPP
 - *Docker (Virt. SO) Kubernetes*

- Otros: *NFS* (*Almac. Distrib.*) o *SSHFS* (*Mount* remoto), *Monit*. Red como *logcheck* (envia email a Admin. si acceso no rutinario), de *Servic..* o Servidores de *DNS*, de *Servid. BD (SQL)* o de Correo *SMTP*
- *Netware* de Novell (no confundir con *Empresa Netscape (Mozilla)*): (*Hist. SO*) dejó de soportarse en 2009 pero de los primeros *SO Servid.* que permitian *Compart. Recurso* en red (*Impr., Archivos...*) para *Sucursal* Bancos, Hospitales o Coopraciones. Los *Client.* puede ser variados (Linux, Win,...). Ya en 1993 tenia un *Servic. Dir. X.500 LDAP* (6 años antes que el 1er Active Dir. de *Win*). Gracias al uso de *Pila Protoc. IPX/SPX* (similar a AppleTalk y NETVEUIS (MS)). *Hist. R.*)
- Free*BDS* (Berkeley Software Distribution *Libre*): *SO Servid. Unix* para *Servid. Web, S. Email,...* con *Servic.* robustos (soporta miles de *Proc.* de *Usr*). Sus *Jail* permite hacer *Virt. SO*. Ver *OpenLDAP*
- *Solaris* (antes de Sun ahora de *Empresa Oracle*): *SO Servid. Unix BDS (FreeBDS)*. Sus *Zones* permiten *Virt. SO*. Ver *Cluster*

47.8 Servicio de directorio: estructura jerarquica

- *Servic. Dir.* o de Nombres (*Servic. Dir.*): Permite *Admin.*istrar los Recurs. (decir que *Usr/Grup.* los usan) de forma *Segur.* (con *Autent.*) y *Remot.* (desde un PC o *Servid.* central con un *SO Servid.*). Es un componente esencial de los *SO Servid.* que *Map.* los muchos *Recursos u Obj.* de Red (ver abajo) a sus *Id.* de Red.
- *Obj.* o *Recurso Red*: son *TAP*. Cada uno tiene su Nombre o *Id. Unic.* y *Atrib.*, al ser tantos se *Estr. Jerarq.* en *BD* de *Dir.* o *BDOO* (Esp. Nombres *Domin.* y *Ambito*).
 1. *Fich.:* carpeta *Compart. Samba* (o *FTP*). El Admin. garantiza *Disponib.*, *Backup* y *Segur.*
 2. *Softw.:* *Despl.* y *Actual.* de Apl. Uso *Remot.* de Apl. del *Servid.* o *Sincr.* de Apl. (*Cluster*). *Servic.* de *Email*.
 3. *Perif.:* control de *Acces. Segur.* a PCs *Distrib.* u otras redes: *Impr., IoT, Faxes*
 4. *Usr/Grupos:* Guardar Inform. *Telef., Emails,.... Autent.* *Segur.*
 5. *CPUs* como *Cluster*
- *LDAP* (Lightweight Directory Acces. Protoc. ∈ X.500 Protoc.): especificado en *RFC*, usado por Active Dir. y OpenLDAP (FreeBDS). *Estr.* en *BD Dir. (Jerarq.)* la info de *Obj.* (*Usr, Password, nombres, Telef. o Apl...*) y permite *Compartirla* dentro de una *Intranet* o *Internet*. Funcionamiento ver (*Sesion Active Dir*). Puede ser usado por *Veyon* Ver *EtcPasswd EtcShadow Netware*
- **Active Dir.** Domain Service (AD DS, *Primo Juan*): programa de *Win Server (SO Servid.)* que corre en el *Servid.* (llamado *Controlador de Domin.*) y que usa *LDAP*. Permite *Admin.* los 4 *Obj.* de una red organizada en **Domin.** y **Manten.** (resolver incidencias)
 - Inic. *Sesion:* de *Client.* ⇒ conecta a un *Servid. LDAP* usando dicho protocolo.

- *Estr.*: **Domin.**: es un conj. de *Obj.* de red o *Client.-Servid.* con Group Policy de *Segur.* comunes. *Arbol*: conj. de Domin. *Jerarq.* *Bosque*: marcan los límites de relac. de config. y no los Domin.
- *Samba*: hablar de que corre en *Linux* o *Win*.
- *Controlador Domin.*: *Servid.* de Windows que contienen la *BD* de A. D. y ejecutan funciones relacionadas con AD, como la *Autent.* y la autorización.
- Rel. de Confianza: es un Sist. Global de *Acces.* en *Win Server* para *Autent.* Usr fuera de *Domin.*
- ACL (ver *BD* de *Autent.*)

Ver *Petic.* *IRP Win.*

47.9 Almacenamiento Distribuido

- *Almac. Distrib.* o (*Almac.en* de *Dat. Distrib.*): donde la Inform. es *Almac.* en más de un *Nodo*. Puede referirse a una *BD Distrib.* (MongoDB o Apache Cassandra) o a un sist. *P2P* (*BlockChain*).
 - *Tma CAP*: en Almac. Dat. Distrib. solo 2 de los 3 se pueden garantizar a) *Consist.*, b) *Availability* (*Disponib.*) y c) *Part.* (tolerancia al P.) Ej: ver *BD Distrib.*
 - *Sist. Arch. Cluster* (Clustered File System): puede usar uno de los de abajo a, b, c o d. De forma similar los *Array de Disco* (ver *Redund.* RAID)
 - a) *Sist. Arch. Distrib.*: *NFS Samba* abajo
 - b) SAN (Storage Area Network): permite Acces. Directo a *Dispos.* de *MS* distantes en *Form.* de *Block Mem..* Difer. por *Ext. Red PAN, LAN WAN*. Ver Fig. 47.15
 - c) NAS (Network-Attached Storage o Almacenamiento *Conect.* en Red): provee Almac. y Sist. Arch. con protoc. como *NFS* o *FTP*. Suelen estar dispuestos en RAID (*Redund.*).
 - d) DAS (Direct Attached Storage): conexión clásica directa al como PC SATA

Ver *Container (Virt. SO), Nube, Met. Acces. (SAM), VFS, Nube.*

- *Sist. Arch. Distrib.* (no confundir con *SSHFS*, ni *Arch. Compart. (Ln)*): tipo de *Almac. Distrib. Cluster*
 - ***NFS*** (Network File System, no confundir con *Sist. Arch. NTFS*, ni con *VFS*): *Sist. Arch. en Red (Almac. Distrib.). Protoc.* del *IETF* para que *Linux* acceda a otros *Sist. Arch.*, para *Compart.* o *Mount Dir.*ectorios en Red. Es de Sun Microsystem *Oracle*. Ver *Samba NAS* (abajo)
 - ***Samba*** : es un *Sist. Arch.* de Red (*Almac. Distrib.*) *Impl. Libre* del Protoc. SMB (Server Message Block) o Common Internet File System (CIFS, renombrado así por MS *Win*). Corre en *Unix* permitiendo que se vean como *Client.* o *Servid.* en Redes *Win Integr.* o *Unif.*icando ambas redes. Permite *Compart. Recursos Red* como *Dir. (Fich.)* e *Impr.* (e *IoT??*). Además soporta Active *Dir* (*Servic. Dir.*). Ver *FTP, SSHFS, Activ. Dir. NFS*

47.10 Softw. red: monitorización y gestión

- *Monit.* y auditorias: *SNMP*, Pandora FMS, *logcheck* (*Ubuntu*). para *Admin.* los 4 *Obj.* red (*Servic. Dir.*).
- *Simul.*: *CISCO* (*Packet Tracer*)
- MicroSoft *Proyec.*
- *Stack Soluc.*: como IIS

47.11 Softw. red: aplic.

- Apl. *Client.*: *Naveg.ador* (*Chrome*), Client *Email* (*Thunderbird*), Escrit. *Remot.* (*NX-Client*), Gest. Descarga (*Filezilla FTP*).
- *Driver* vs *Firmw.* de *Tarj.* Red o *Rut.* (*CISCO IOS*)=*Traduc.*
- Apl. *Web*, Servid. *Web* Servid. *BD*,
- Otros: *Protoc.*, *Socket*

47.12 Softw. red: pila de protocolos

- *Pila Protoc.* y Tab. *TCP/IP vs OSI* (práctico vs teórico, *Encapsul.*)
- Fig. 4 ej. Protoc. por capa(Tab. *TCP/IP vs OSI*)

47.13 Kubernetes

- *Kubernetes*

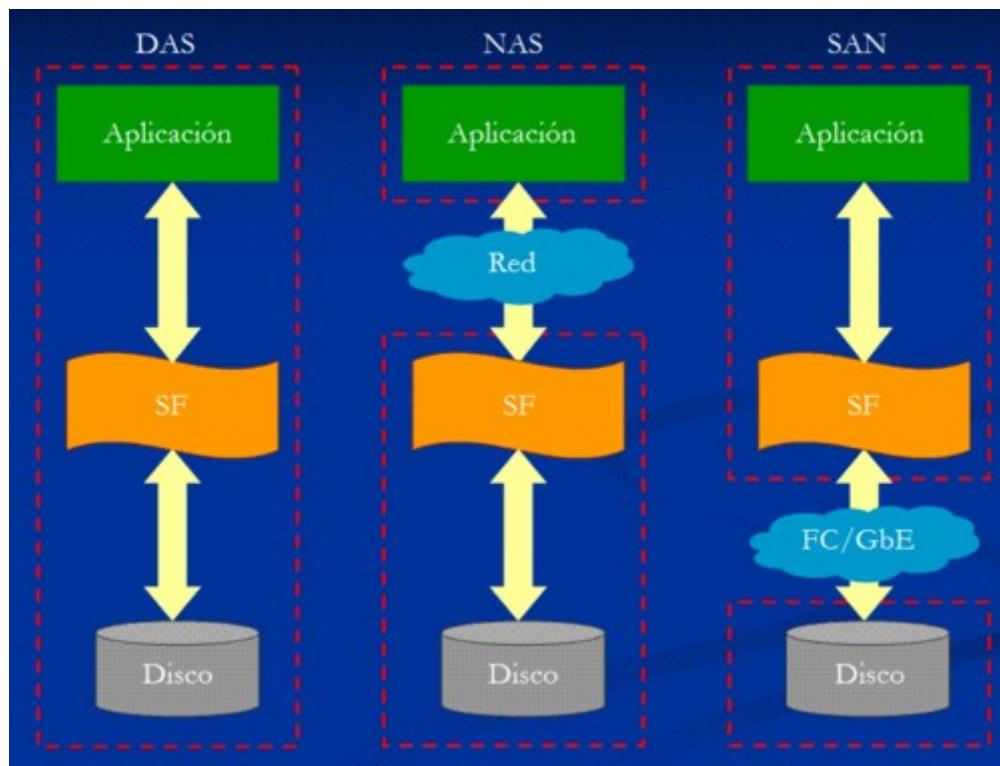
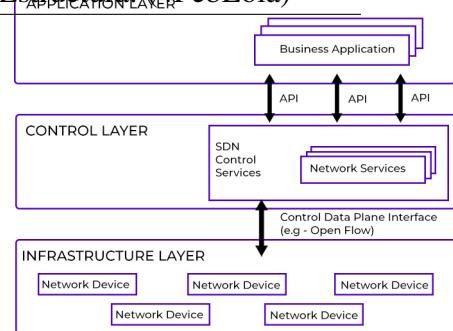
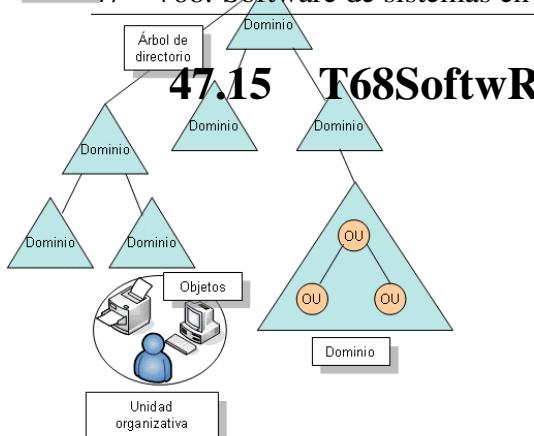
47.14 Otros: remoto, teletrabajo

- *Admin. Remot.* : admin. desde *TeleTrabajo*:
 - *Admin.*: *SSH*, Active Dir. (*Servic. Dir.*)
 - Escrit. R.: NX Cliente/Server con Gestor de Ventanas X11 (*GUI*). *Veyon*, *Servic. E. R.* (*Win Server SO Servid.*)
 - Bossware: jefe espia (*Malw. Spyware*)
 - Otros de C. *Sesion*: *SSH*, *SSHFS* (ver). *Veyon*

Ver *WakeOnLAN*, *Commit Git*

- *Veyon* : *Admin. Remot.*, puede conectarse a un Servid. *LDAP*
- *SSH*.

Bosque 47 – +68. Software de sistemas en red. Componentes. Funciones. Estructura. (4FebLola)



Chapter 48

*+69. Integración de sistemas. Medios de Interconexión. Estándares. Protocolos de acceso a redes de área extensa (7EneCarmenCampos)

48.1 Introducción. Conclusión

- Ej: ¿Qué es un *Switch* vs *Rut.*?
- ¿Como *Almac. Distrib.* como *Samba Integr.* distintos SO?
- ¿Como comunicar muchos hardw. (*IoT*) y subredes de forma *Transpar.*?
- Hist.: *ARPANET*, *RED UNO PDN*, *HDLC PPP*, *Roaming (Movil)*.
- Futuro: *GSMA Open Gateway (Movil)*, *NFV (F. en Virt. SO)*

48.2 Arquitectura TCP/IP y protocolos

- Sist. en *Red*
- Arq. Red, *TCP/IP vs OSI*, *PDUs*, *Encapsul.*, *Protoc.*

48.3 Integración de sistemas en red

- **Integr.** acción (no confundir con *Tecn. I. Transistor (3D HBM)* vs **Integridad** en *Segur.*, *Fich.*, *BD (Restr.)*, *I. Ing.* *Softw.* (*Cooper.*)): combinar diferentes *Compo.* Hardw. y Softw. en una sola *Red* para comunicar nodos *Transpar.*, Es componer sub*Redes heterogeneas* para que *Interop.* gracias a *Modul.* de *Pila Protoc..* En *Guerra Protoc.* surgieron varios (*TCP/IP,...*) y la *ISO* sacó *OSI* para *Interop. independientemente del fabricante*. Ej: *Servic. Int.* (abajo) Ver *Segur.*, *Checksum (Err.)*, *Modul.*, *IDE*, *IoT*,

48.4 Proveedor, tier1 y PDN

- **Proveedor** o **Telco** (*Telef. Company*): *Empresa que da Servic.* de *Telecomun.* a sus clientes.
 - **Tier** (Unidor, from tie): *Capa de infraestructura de Cable de Internet o de PDN.* Fig. 48.14.
 - Tier1: de *Empresa u Op.* de grandes países (ATT, Telefónica, Deutsche Telekom). Se comunican gratis entre ellos y le cobran a los Tier2 (que no les hace gracia) y estos a los Tier3.
Aunque las Tier1 se quejan de las *Tecn.* no les pagan (*Google...*) ver GSMA Open Gateway (*Movil*).
Ej. Lumen posee la Red T. 1 internacional (normalmente Cables Submarinos) más larga 900000km (Lumen no paga por usarla). ATT 700000, Telefónica 65000.
 - Tier3 o **ISP** (o *Internet Servic. P.*): da Internet en *Home* mediante *ADSL, Fibra, Movil,..* Construyen *WAN*

Ver *BD Nube, Instal., PPP, PDN, GSMA Open Gateway (Integr. Moviles)*

- **PDN** (Public Data Network, *Red de Dat. Public.*, o Propietaria??): operada por un *Proveedor Priv.* para dar *Servic.* al público. Forman la infraestructura de *Internet*.
 - **Red UNO** o RETD/Iberpac: de Telefónica 1972 y la canadiense DATAPAC 1976, basadas en X.25, fueron las 1a redes *Conmut. Paq. Hist. R.*.
 - Red Propietaria (no confundir con Red *Priv.* o *PDN*): *Cables Backbone* submarinos o de *Fibra* entre Proveedor (*ATM*) o *WAN Universitarias*.

Ver *VPN, Kerberos (Cript.)*

48.5 Hardware de red

- **Hardw. Red** : *Compo.* de *Nodos Dispos..* Muchos de *CISCO* simulables con *Packet Tracer*. Distinguimos:
 - **ETD** (Equipo Terminal o Extremo Dat.): *Nodo extremo de comunic.*
 - a) *Host Servid..*
 - b) *Host Client.* (*Estac. Trabajo, Impr. Movil*)
 - c) *Perif.*: *NIC* y ciertos *Transcep..* Ver *E2E*
 - *Medio Transm.* : *Wireless* y los 3 *LAN*: sobre todo *Cable Estr.* como *Par Trenz.* y *Fibra FTTH Instal.* radio < 3 cm. *Repet.* y *Splitter*
 - **ECD** (Equipo Comunic. o Conectividad Dat.): Nod. intermedio que *Exten.* Red (*Recurs.*) organizados por *Capas* (Tab. y Fig. 46.13). Ej: *Modem, Hub o Switch.* y *Gateway Residencial (Home* que engloba a varios).
 - **GPON (FTTH)**: distingue *ONT* y *OLT*.

Ver *C. Fisica. CISCO*

<i>Capa</i>	<i>Dispos.</i>
<i>C. Fisica</i>	<i>Hub, Modem, Repet.</i>
<i>C. Enl.</i>	<i>Switch, WAP</i>
<i>C. Red</i>	<i>Rut.</i>
<i>C. Transp. C. Apl.</i>	<i>Gateway</i>

48.6 Medios de interconexión en capa física

- **Modem** (Modulac.-Demodulator): en *C. Fisica*, *PC(Digit.)* → *Modem(Analog.)* → *RedTelefComm* i.e. conversor de *Signal Digit.* a *Analog.* que permite T/R en *Red Telef. Comut.* (tipo *DSL*).
Exprime el A. *Banda* (como aprovecha *PLC Red* y F. *Multiplex*.)
 - Actual.: *PC* → *Modem* → *WAN* i.e. cualquier *Dispos.* que permite *PC* acceder a *WAN*, ya que hoy las Comunic. *E2E* son *Digit.* y no es necesario Modular. a *Analog.* Ej: *Gateway Residencial (Home)*, M. *ADSL* o M. *Cable (HFC, ATM)*
- **Hub** (Concentrador): en *C. Fisica* mensaje que recibe mensaje que envia a todos los *Hosts (Broadcast)*.
No *Filtr.* por MAC como el *Switch*.
En desuso por baja *Segur.* y alto A. *Banda*.
En *Ethernet 100BaseT* actua como *Medio Compart.* de *Red Bus* (o mismo Domin. *Colis.*).
Sin *Colis.* solo 1 dispos. puede transmitir. Los dispos. deben encargarse de detectar *Colis.* y si pasa hacen una pausa para luego volver a intentarlo.
El *Switch* lo soluciona pudiendo hacer Broadcast también. Ver *Colis. CSMA*.
 - *Repet.*: por *Aten. Shannon. Fibra R. Luz.*
 - *Cables Backbone LAN* y *WAN*
 - *PLC Red* y *HyFi*

48.6.1 En capa de enlace

- *Switch* vs *Brigde*: para *LAN*.
- *WAP*

48.6.2 En capa de red: routers y gateway residencial

- **Rut.** (Router, Encaminador): en *C. Red*, conecta redes con distinta Direc. *IP* de *Red* (terminadas en 0 en parte de *Host*). Tienen varias *IP*, una por cada *Gateway*!!.. Las F. principales son:
 - Enrutar *Efic.*: con *Tabla* Dinam. (*IGP*: *RIP,...*) o *Estat..* También *MPLS* por Etiq. y no *IP*. Es P *Dijkstra. Busq. Vel.* con Mem. *Asoc.. Conmut. Vel. Impl.* por hardw (AND hardw.)

- Tienen un *Patch*, hoy son maquinones *NOS*.
- *Segur.*: *Filtr. Firewall* por *MAC*, Pasw.
- *Frontera*
- **Gateway residencial (Home)**: *LAN* → *WAN*. confundidos por converg. *Tecn..*

Ver *MPLS*, *Multiplex.*, *WAP*, *PPP*, *Direc.*, *Dir.*, *Rut.*, *Crit.*, *TTL*

- Tipos de *Rut.*:

- Según ubicación en red:
 - a) R. Internos: dentro de la *LAN*.
 - b) R. Externo: comunic. Nod. de LAN con exterior.
- Según Alg. encaminam.:
 - a) Estat.
 - b) *Dinam.*

Otros tipos:

- R. Residential Gateway: para conectar un SOHO (Small Office, Small Home, Empresa PYME) a un *Proveedor* (ISP)
- R. *Empresa*: como Dis. *Jerarq. Red.* según capacidad:
 - a) *Acces.*,
 - b) *Distrib.*,
 - c) *Kernel*
 - d) *Borde*.
- R. *Wireless*: *Interf.* entre redes fijas y móviles (*Wifi*) (QUITAR resto?? GPRS, Edge, Movil UMTS, FrizBox, WiMAX)
- Otras variantes:
 - a) Brouter=*Brigde+Router*: operan en *C. Enl.* y *C. Red* según *Protoc.*

48.6.3 En capa de transporte y aplicación

- **Gateway** (Puerta de Enlace o Pasarela): En general son *Traduc.* entre *Protoc.* i.e. *Interf.* en *Capas altas*, entre 2 *Dispos.* que permite compartir *Recursos*. Hay 2 tipos:
 - G. *C. Red* o Residencial (*Home*): el que realiza Op. *NAT* (*Traduc.* IP Priv. a Publ.). NOTA: en *Packet Tracer*. al *Config.* un PC, es la IP del *Rut.* más cercano donde enviar Paq. IP que no es de mi Subred. Su IP suele ser: 192.168.1.1 o 192.168.0.1 y utiliza algunos rangos predefinidos, como por ejemplo 127.x.x.x, 10.x.x.x, 172.x.x.x, 192.x.x.x.
 - G. *C. Transp.*: conectan redes con diferente *Protoc.* Tranp. (*Protoc. incompatibles*). Ej: TCP/IP con ATM o SNA (*Guerra Pila Protoc.*)
 - G. *C. Apl.*: *Traduc.* *Form.ato* de Apl. Ej: de *Email* a *SMS Movil*.
 - Otras F.: *Firewall* o *Proxy* en redes de *Empresa*

Ver *IGP (Rut.)*

48.7 Resumen de protocolos por: acceso a Internet desde LAN

- **Internet**: conj. de *Redes* (red de redes) que usan el protocolo *TCP/IP* (gana *Guerra Protoc.*) para comunicar nodos (ver ej. abajo). Se basa en *Encapsul.* y *Modul.* (*Filos. Pila Protoc.*).

Es *Filos.* parcheados (ver *TCP/IP*) y *Filos.* democratica tiene el *Probl.* del Mejor Esfuerzo (*QoS*) y no permite [VoIP].

Su precursora es *ARPANET* (cuyo **único requisito pedido era la Robust.** ver *Red Hist. R.* luego es de *Commut.* Paq.. Regulada por la *ISoc* y el *IETF*. Iniciado con “Red UNO” *PDN*.

Es una *Red Superp..* Ver *Seis Grados*, Mejor Esfuerzo (*QoS*).

Conect. a I. Actual.: partimos de un *Dis. Jerarq. Red* con Red de Acces. *FTTH* (SONET y GPON):

-) Cliente obtiene su IP (Public. y Priv.) con *DHCP* (ver *Truco DNS*).
-) *NAT* permite distinguir IP *Priv.*, *ARP* averigua *MAC* desde *IP*.
-) La IP del Servid. la da el *DNS*.
-) Este hace *Petic.* a un *Servid.* externo (ej. *HTTP* desde *Naveg.* o *FTP* desde Apl. *Java* o mi Proyect. *FC*).
-) La IP y MAC del Cliente las recibe el Servid. y este responde con lo que pide.

48.8 Protocolos LAN de acceso a WAN y de integración: PPP

- **PPP** (Point To Point Protocol): en *C. Enl.*, conect. *E2E 2 Rut.* directam., sin ningún *Host* intermediario.
 - a) Requiere que solo sea *Duplex*.
 - b) Puede proporcionar: *Cript.*, *Compr.* y recuperación de *Err.*, *Direc.* *IP Dinam..*
 - Actual. se usa en SONET (*FTTH*) y en *Telef.*, *Movil* [PPP]. Mas marginalmente en:
 - a) PPPoE (PPP over *Ethernet*, aunque raro algo usado en GPON *FTTH* [PPPoE]).
 - b) PPPoA (over *ATM*).
 - c) PPTP (Tunel *VPN C. Sesión*).
 - Ha permitido a los ISP (*Proveedor de Internet*) establecer linea *DSL* al cliente y dar *Servic. Int.* (*ISDN Hist. R.*)
 - HDLC (High-Level Data Link Control): *Hist. R.*, originó el *PPP* y deriva del SDLC (*IBM SNA Guerra Pila Protoc.*)

48.9 Protocolos en capa de acceso

- *IEEE 802*: de Facto *Bluetooth*, *Ethernet*, *Wifi*, *WiMAX*,
- *HFC* (Fibra+Coax.), *WiMAX*, *Movil*,

48.10 Protocolos: En capa de red: VLAN y STP

- *VLAN*: IEEE 802.1Q. Permite al *Admin.* agrupar Hosts incluso sin estar conect. al mismo *Switch* (Fig.). Hasta 1989 no podíamos conectar (*Integr.*) varias LAN (los *Rut.* eran caros). En 1990 llegan los *Switch* y si podemos.
- STP *Recubr.*: soluciona problemas de *Circ.* bucles en *Ethernet*

48.11 Protocolos de integración

- Protoc. a WAN: *Servic. Int.*, *MPLS* (filosof. de Integr. y antidemocratica)
- Antiguos: *Telef.* (*DSL* o *Servic. Int.*), *X.25*, *Frame Relay*, *ATM* (*Ethernet WAN??*)

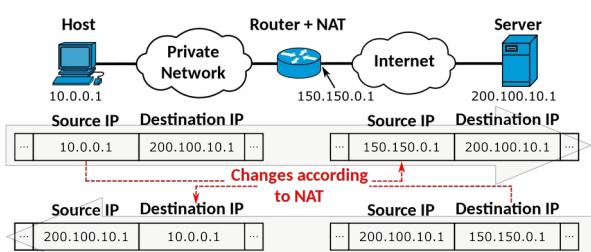
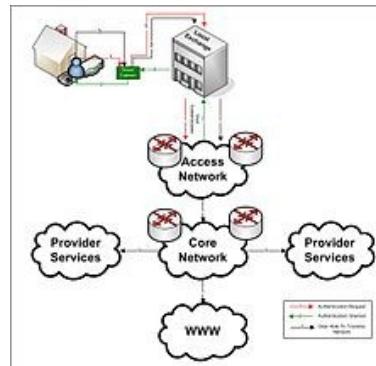
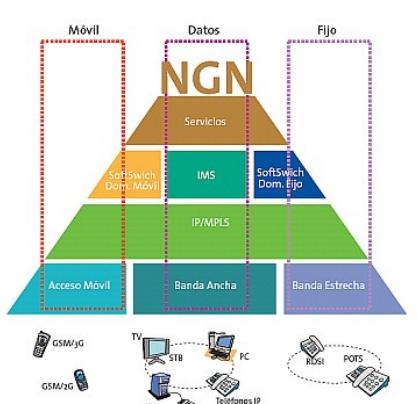
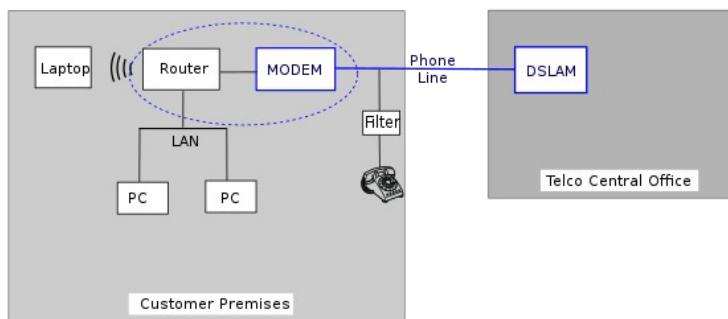
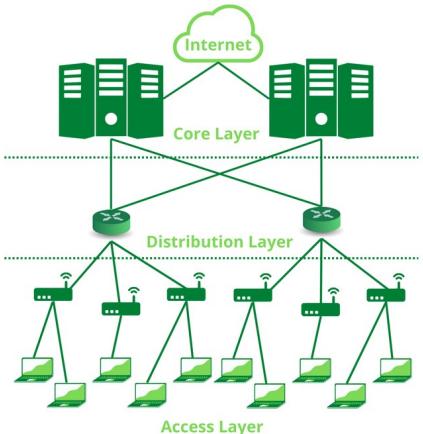
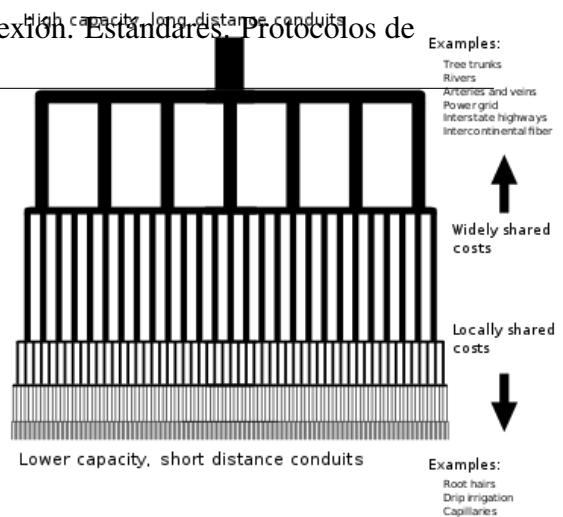
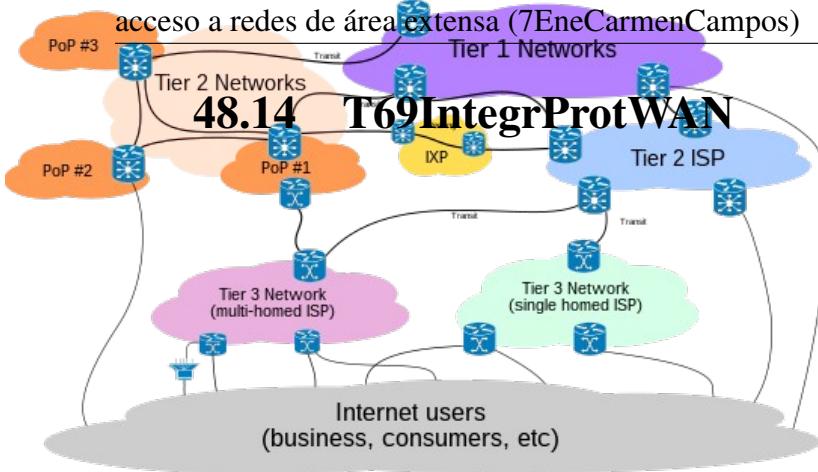
48.12 Protocolos en Capa aplicación

- *Samba* (integr. SO)
- *VPN*

48.13 Ejemplos de LAN a WAN

- *Red Superp.*, *ExTranet*, *Sucursal*

48 – *+69. Integración de sistemas. Medios de Interconexión. Estándares. Protocolos de acceso a redes de área extensa (7EneCarmenCampos)



Chapter 49

+70. Diseño de sistemas en red local. Parámetros de diseño. Instalación y configuración de sistemas en red local. (12EneAlex)

49.1 Introducción. Conclusión

- Ej. Motivador: dado un Presupuesto y unos Servicios a cumplir ¿como diseñar la red?
- Ej. Ver servidor de MineCraft de Antonio: PPTP, Configurar WAN, DMZ.....
- Ej. *Centro Dat.*
- Parte de *Admin.* de Red T71??
- Hist.: ARPANET antes no requería *Segur.*, llegada de los *Rack*
- Futuro: importancia del *Admin.* IT (FP ASIR), *Energ.* Prop. Comput. (*Ecolog.*)
- Actual: *Centro Dat. Satelite*

49.2 LAN

- *Wifi* y *Ethernet*

49.3 Diseño de sistemas en red local.

- *Arq. Client.-Servidor, P2P*
- *Dis. Jerarq. Red* (Hierarchical internetworking model): *Model.* de *Red* con 3 *Capas* o *Tiers* (Fig. 48.14). Propuesto por *CISCO* y basado en dominancia de *Ethernet*. Para *InTranet* de *Empresas* (ver abajo). Alternativa a *TCP/IP* (paso de 4 a 3)??

- Capa *Acces.*: cada *Usr* con su PC por *Switch* y *Ethernet*. En la Fig. 48.14 vemos que tanto un PC como un *Móvil* acceden mediante la red de “último kilómetro” (last mile), Ej: **FTTH, ETTH** (usar *Ethernet* para llevar I. al Cliente).
- Capa *Distrib.*: conecta ambas capas según una política. Varias *Sucursales* conectadas en *WAN*.
- Capa Core o *Backbone* o *Kernel*: donde está el *Gateway a Internet* o *Rut. Frontera*, comunic. a alta Vel.

Ver *Pila Protoc.*, *Tipos Rut.*, *Internet*.

- **Frontera** (Customer Edge Router, no confundir con *Frontend*): *Rut.* el que le da *Internet* al *usr* y se conecta al *Proveedor ISP*. Hay *Rut.* del F. Proveedor y R. del *Usr*. Pueden usar protocolos tipo: *IGP* como OSPF o External GP como BGP o de Label como *MPLS*. Se hace Comput. *IoT* F. (ver).

49.4 Parámetros de diseño.

- *Proyec.*,: KOrganizer y MSProject,
- *Opt., Rut. Crit.* (*Diagr. Gantt*), *Precio, Escal.*
- *Traf.* (*Flujo, T. Colas*), *Topol.* (*Geom.*) y *Segur.* (*Backup, DMZ Milit.*), *Servic.*

49.5 Instalación de LAN

- **Cable Backbone:** *Medio Transm.* principal en *Red Bus* donde debemos cuidar la *Congest.* (*Fulkerson*). En *LAN* como pisos es **C. Vertical** que une los *Racks* o *Patchs* de las plantas (Fig. 49.10). En *MAN* o *WAN* son los C. *Submarino Tier1*, también *FDDI* (*Fibra*) debido a su alto A. *Banda servia??* para *LAN* y *WAN*.
 - C. *Estr.* o *ISO/IEC 11801*: principalmente **Latiguillos Par Trenz.** UTP o STP para *LAN Ethernet* aunque también *Fibra*, *Patch Panel* y *Conect.* clásicos. Para *Servic. Int., IoT* (fábricas). Define varias categorías de P.T. (A,B,..,F) y de *Fibra* (OM1,..,OM4) que se corresponden con los CAT clásicos.
 - C. **Submarino**: *Backbone* de *WAN* normalmente de *Proveedor Tier1*. *Integr.* redes. Antes de Cobre, hoy de *Fibra*. El 95% del *Traf. Internet* va por el [Roboticus]. Ver SONET/SDH (**FTTH**)
 - *Instal. Segur.* de C.: ver *Tierra*
 - Ley L. de Infraestructuras comunes de *Telecomunicaciones (ICT)*: para interior de edificios *Empresa PYMES..* como *Instal. Cable* y *FTTH*

Ver *Control, Mod. Acces., Disponib., Servic. Int., PLC Red, Tierra, C. Modem, Electr., Wireless*

- **Tierra Electr.:** lugar donde confluyen todas las Intensidades de *Cables*. *Instal. Segur.:* si por los pies del humano pasa Intensidad que va a T. los Fusibles la *Interr.*

- *Instal. Segur.* de *Cables*: CUIDAR: No *Interf.*, T. al mismo nivel. Volt. OK y sin fugas. *Conect.* en zonas *Acces.* pero Seguras.

Ver *EM, Apantall. Par Trenz., Coaxial, Diafon., Interfer.*

- *Repet.* : en *C. Fisica*, para *Exten.* red (por *Aten.* de *Pot.*) (genialidad *Shannon*). Se dice R. si *Analog*. y **Regener.** si *Digit.* (*Signal Bits, C. Fisica*). A veces son *Transcep..*

- *Splitter* de *Luz* es lo contrario (GPON FTTH). Hay R. *Luz* Opticos sin electr.! (*Fibra*).

Ver *Topol.* Red Anillo, *Colis.* CSMA.

- *Herram.* de *Instal.*: poner los elementos con *Segur.* y fácil *Acces..* Para *Certif.* que todo OK.

- *Par Trenz.:* Crimpadora (RJ45) Puncher (rosetas, ver *Tarj. Perforada*), *Tester* y *pelaCables*

- *Fibra:* fusionadora, cortadora

- Otros: Multímetros (medidor de *Signal*), osciloscopio. Funda **Termoretractil** (al calentarse aprieta los *Cables*), bridás, *Id.*, Destornilladores, Allen...

- Otros: H. *Proyec.* (KOrganizer y MSProject), H. *Monit.* (Pandora...), CASE, IDE

- *PLC Red* (Power Line Communication, Comun. Lin. *Pot.*, no confundir con PLC *Autom.*): conex. por *Cable* cobre de red *Electr.* (como *Modem Multiplex.* en F.) que va a un frecuencia más baja (50Hz). F. como *WAP* y *Repet.*

- Sist. HyFi: combina *Wifi+Ethernet+PLC Red*, optimiza uso de las 2 redes caseras (electr. y datos)

- **FTTH** (X) (*Fibra To The Home*, Premises): casi todos en España la tenemos (*Ley Infraestr. ICT*). *Instal.* complicada en casa por dobleces de esquinas con radio > 3 cm [Colibrí] según *ITU-T G.657* (est. para SMF, monomodo cara??). **Cortar perfecto** para fusionadora de 2000 Euros.

- GPON (Gigabit Passive optical network): basada en *Splitter* de *Luz* y Opt. Line Terminal (OLT) y Opt. Network Terminal (ONT) ver ??

- **SONET/SDH** (*Sincr. optical networking and synchronous digital HJerarq.*): SONET se usa en America y SDH en resto. Usa ambas *Multiplex.* TDM y en WDM. Protoc. con corazón *ATM* y usado por *Proveedores* para *Cable BackBone Fibra* en Redes de *Conmut. Telef.* y de *Servic. Int..* Usa *PPP*.

Ver *HFC=Coaxial+F. (Triple Play)*. ETTH (*Ethernet To The Home*).

49.6 Instalación: centro datos

- *Centro Dat.* : para una de las 3 *TAP* compuesto por *Racks de Servid.* o *Patch.* *Req. Disponib.* (*Redund.* SAI, RAI) e *Instal. Segur.* (las 4, *Temp.* ... ver). Con *Gateway*, *Balum Transcep..*

- C.D.: se transladan al espacio (*Satelite*) para ser más *Ecolog.* [ElPaisDic22]

- **Centro Respaldo Datos:** toma control del C.D. en caso *Fall.* de Red. Es como *Backup* o *Disponib.*
- *Ecolog.:* consumen mucha *Energ.* por la *IA* online o *Blockchain*

Ver Alto *Rend.*, *Nube*.

- **Rack** (Bastidor o Armario de *Distrib.*): metálico (similar a Carcasa de PCs *Estand.* a 19 pulgadas de ancho) con ECD. Suelen tener *Servid.*, *Switch*, *Rut.*, *Patchs* o *Firewalls*. Debe ventilar para *Temp.* (*Instal.* *Segur.*), *Electr.*. En *Centro Dat.*
- **Patch Panel** (Panel de Parcheo): receptor de todos los *Cables Estr.* que organiza *Conect.* en *Switch* o *Rut..* Conect. por *Backbone* (*Cable Estr.*) y de forma *Segur.* (ver *Tierra*). Hay que *Docum.* la *Id.* o *Etiq..* Ver Fig. 46.13. En *Centro Dat.*
- *Almac.* *Distrib.*, *Cluster*, *Backup*

49.7 Configuración de LAN: componentes

- **Home** o *Gateway* Residencial (o doméstica, hogar): *Hardw. Red LAN* que *Integr.* *LAN* → *WAN* Fig. 46.13 [Router]. La Converg. *Tecn.* hace que no sepamos que *Compo.* tiene [*TecnConverg*] pero suele tener:
 - *Modem*:
 - *Firewall*:
 - *Rut.*, *Switch* y *WAP*. Su *IP* para enviar a *Internet* suele ser 192.168.1.1
- **WAP** (Wireless Access Point, o AP Punto de Acceso, no confundir con *WPA*): permite a Dispos. Wireless (o *Cable*) conectarse a una red Cableada, haciendo de *Integr.* en *Topol.* Red Estrella o Infraestr. Puede o bien estar *Integr.* en un *Rut.* o bien conect. directamente con un *Rut.*
 - *Hotspot*: es una ubicación física en la que hay acceso *Wifi* disponible. El *Hotspot* se crea usando wifi.
 - Botón WPS: de *Config.* ver *Vulner.* *WPA*
- **NIC** (Network Interf. Controller o Tarj. Red): Ofrecen *F. MAC* y *LLC* en *C. Enl.*: preparan y envian/reciben datos al *Medio de C. Fisica.* *PC(SOyDriver – ComoALSA)* → *Tarj.Red(Firmw.)* → *Red* i.e. su *Firmw.* permite al PC mediante el *Driver* comunicarse con una Red cualquiera.
 - *DMA* e *Interr.* ofrecido por NICs modernas.
 - *MultiHomeing*: es conectar un PC a más de una red, por si una de las 2 *Fall.* (*Redund.*)
 - *Dual-Homed Host*: tipo de *Firewall* que Filtr. entre una Red *inSegur.* y una segura. El Sist. suele estar compuesto por 2 NICs (*Ethernet*), pero también un *Proxy*, *Firewall*, *Gateway*, cualquier *Servic.* que Filtr. Ej: **Screened Host**: un *Rut.* + *Host* donde se ejecuta un *Proxy*.

Ver *Filtr.* *MAC*, *Topol.* Red Ad-Hoc. Dual-Homed (*Firewall*)

49.8 Configuración de LAN: enrutamientos

- **CISCO** (poner en T70): *Empresa de Hardw. Red y Manten.* como: CISCO IOS (*NOS*). Famoso por Packet *Tracer*. Ver *C. Acceso, Mode, Jerarq. Red*
- Packet **Tracer** (Rastreador de *Paq.*): *Simul.* una Red *CISCO*, implementando su *Topol. Red*. Se *Config.* *Visualmente* y luego cada *Rut.* con *CLI*. Conocerlo para tener *Certif. CCNA*. Las tareas de *Config.* y *Admin.* son:
 - *Asign. IP Estat. y Masc. a Hosts*
 - *Asign. IP Estat. a las Interf. del Rut.* (los *Switch* no necesitan)
 - *Asign. Gateway a Hosts* (también debería *DNS*) y si no *DHCP* (*Config. Autom.*)
 - *Config. Tabla EnRut. Est.* (mandar paq. al Nodo (*Switch* o *Rut.*) que siga por el camino más corto).
 - *Test Ping*
 - *Asign. Nombres: a Wifi (SSID) y Host de red.*
 - Otras: *Instal. Driver Tarj. Red.*

Ver *Comando Tracert=Traceroute* (como *Ping*), *Backtracking*, *Comando Tracert (Ping)*

- RESUMIR ESTO:
 - *Config. de Acces. a WAN:* del *Rut.* o *Modem*
 - *Rut.:* a) Tareas de *CISCO Packet Tracer* b) *Segur.* c) Op. *NAT* para ahorrar IPs
 - *Gateway:* *Traduc.* protoc. incompatibles
 - *Firewall: Filtr.*
 - *Wifi e Wireless Segur..*
- *VLAN* y *STP Recubr.*
- **DHCP** (*Dinam. Host Configuration Protocol*): un *Servid.* asigna parámetros de red *Autom.* como *IP Dinam.* (ver *Config. Tracer*). *Truco DNS* para tener *IP Estat.* Antonio: al config. *WAN* parece que puede elegir *PPPoE* o *PPTP* (*Tunel VPN*)
- **DNS** (*Domin. Name Server*): usa una BD *Jerarq.* y *Distrib..* Permite a Usr trabajar con nombres organizados en Domin. en vez de recordar IPs. *Config. Packet Tracer Ej: WINS* (de *Win*). Las IP DNS que se suelen *Config.* son 8.8.8.8 (*Google* primaria) y 1.1.1.1 (secundaria).
 - *Truco DNS:* evita usar Servic. como Dynamic DNS y No-IP para tener una *IP Estat. Virt.ual.* Permite conectarte a un *Servid.* casero con IP *Dinam.* (*DHCP*). Consiste en tener un Domin. en *Internet* como 000webhost.juan.com con ficheros como index.php https://www.youtube.com/watch?v=4bJUZPYroBg.
 - *Comando: nslookup*

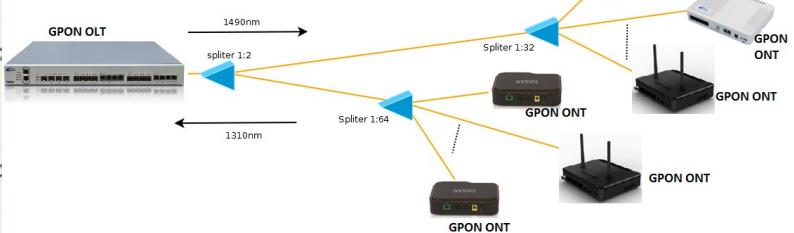
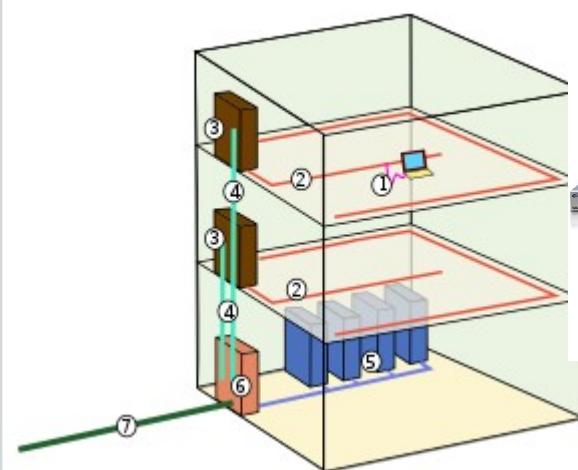
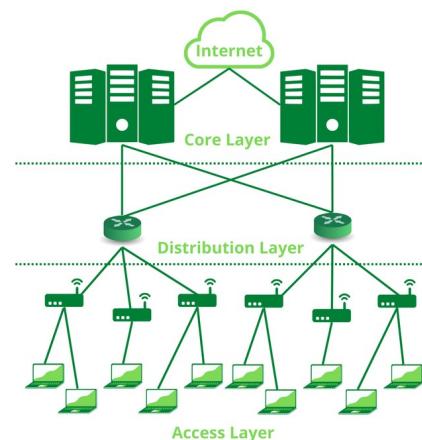
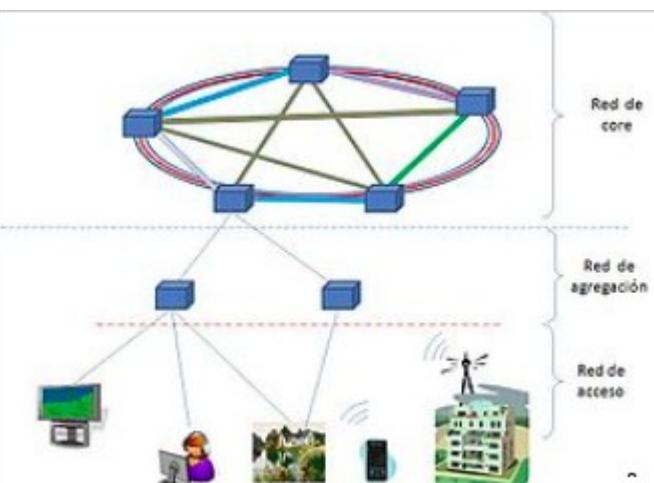
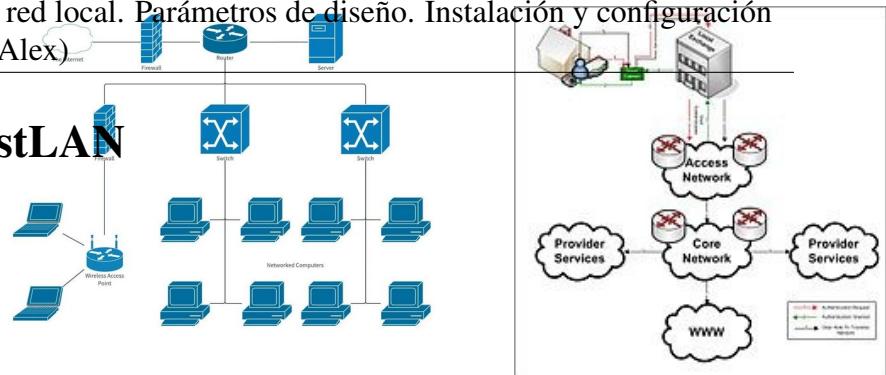
Ver Esp. Nombres (*Ambito*), *ICANN*. *Pharming (Suplant.)*

49.9 Sistemas en red local

- *IoT*
- *Red Superp.* (Overlay Network, no confundir con *Circ. Virt. (TCP)*): *Red Virt.* enlazada con nodos lógicos, usando el soporte físico de otra Red Subyacente (*UnderLying*).
 - Ej: *VPN, Internet* (red de *LANs*)
 - Ej: *VoIP: Telef.* sobre *Internet*
 - Ej: *LANs* en *Internet: Switch (MAC)* con *Rut. MAC*. Los SO son ligeros y comparte recursos entre ellos.
 - Ventaja (Resil. a *Fall.*, que equipos físicos se apaguen). Desventaja (*Latencia*).
 - Ej: Red *LAN Internet* con *MAC* (ver abajo)
 - Otras: *P2P, IoT, VLAN, Circ. Virt.*

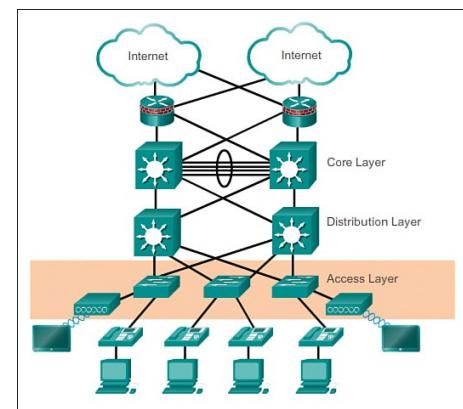
Es un *Unif.. Ver Biolog.*

- *Sucursal* (Branch Office): secciones de una *Empresa* separadas geográficamente pero que quieren *Cooper.* y *Comunic.* (con *VPN o Nube*). Contratan *Proveedor*. Ej: S. Banco Ver X.25, *Netware, Transac., CICS*
 - SET (Secure *Electr.onic Transac.tion*): basado en X.509. Reemplazado por 3D Secure en *Tarj. VISA*.
- Ex*Tranet* (no confundir con *Intr.*): parte de la Intranet (solo los miembros de una organización tienen *Acces. Autent.*) de una organización que se *Exten.* a usuarios fuera de ella, usualmente utilizando *Internet* y sus protocolos.
Similar a una DMZ (*Milit.*) pues se usan *Servic.* desde afuera.
Se suele hacer con *VPN*. Ver *LDAP, Dis. Jerarq. Red*
- *Manten., Testear, Docum.*



Subsistemas del cableado estructurado:

- 1- Cableado de área de trabajo
- 2- Cableado horizontal
- 3- Cableado de administración (armario de cableado, *rack*)
- 4- Cableado vertical (central, *backbone*)
- 5- Centro de cálculo
- 6- Cableado de equipamiento (armario de entrada al edificio)
- 7- Cableado del campus (acometida, cableado entre edificios)



Chapter 50

71. Explotación y administración de sistemas en red local. Facilidades de gestión.

50.1 Introducción. Conclusión

- Packet Tracer (CISCO)
- Futuro: Kubernetes (Admin. Servid.)

50.2 LAN

- Wifi y Ethernet

50.3 Explotación de LAN

- **VLAN** (Virt. LAN): Permite al Admin. agrupar hosts incluso sin estar conectados al mismo switch (Fig.) gracias a switch Nivel 3 (ver). Colección de computadoras en una o varias LAN que se agrupan en un solo dominio de broadcast (ver), filtran el tráfico a nivel de segmento (C. Enl. con switch). Esto disminuye el tráfico, mejora seguridad y reduce colisiones en antiguas LAN [VLAN]. Es por switch no por hardware. Un **trunk** es un enlace por donde va tráfico de varias VLANs. Ej: Fig. 46.13 partición por plantas con switch y por departamentos con VLAN.
 - Config. Tracer: en switch 1º creo (*num, nombre*) de las VLANs (1, default), (2, *progr*), (3, *desarr*) (la default se crea por defecto y a la que pertenecen todos los PC al conectarlos al switch). Luego para indico cada FastEthernet a qué VLAN pertenece en el switch. Solo habrá conexión entre los de la misma VLAN.
 - Basado en *Etiq. IEEE 802.1Q*. Permite separar el tráfico.
 - Tipos:
 - a) V. por Puerto: a cada grupo le corresponde un Puerto del switch (es la más

común).

- b) V. por *MAC*: hay que introducir manualmente que MACs forman cada grupo
- c) V. *Protoc.*: ej: el grupo1 usan *IPv4*, g2 *IPv6*, g3 *AppleTalk*,..

– *Evol. LAN (Hist. R.):*

- a) En 1981 *Ethernet* daba 10Mbps y era *Red Bus* y no era *Escal.*, al aumentar PCs crecían las *Colis.* en *Bus Coaxial*. Por lo tanto no podíamos conectar (*Integr.*) varias LAN (los *Rut.* eran caros).
- b) En 1990 lo se inventaron los *Bridge* y luego los *Switch* que permitieron integrar varias LANs *Ethernet* (gracias al *multiPuerto* y hardw.). Sin embargo hacia los *Switch Cuello Botella*
- c) **Sincoskie (VoIP)** inventó las *VLAN Puerto* para aliviar el problema
- d) Hoy se usa el *IEEE 802.1Q* que incluye las mejoras del *IEEE 802.1aq* con *STP Recubr.*

– Transgresión IP (como *MPLS*): de forma centralizada se puede gestionar que Host pertenecen a que red (como *LDAP??*). Es un *Subnetting* de la C. Enl. añadiendo a *Tramas* un *Id.* de Direc. de Subnet según *IEEE 802.1Q*. Este Id. puede ser por *MAC* o *IP*. Por IP se transgrede la Regla *Pila Protoc.* (el *Switch* de C. Enl. no debe analizar *Cabeceras* de C. Red). Aunque por lo general hay *Correspond.* uno a uno entre IP y MAC podemos tener varias Subnet en una VLAN [VLAN]. Ver *Red Superp..*

Ver *STP Recubr.*

50.4 Administración

- *Segur.* y *Autent.. Cript.* (Kerberos), *Backup*,
- *Comandos de Red:*
 - *Ifconfig* :
 - *Ping*:
 - *Tracert Win* o *Traceroute www.google.es Linux*: *Ambito* y *Latencia*. También *Router*
 - *route*: imprime tabla en *Rut.* (*IGP*)
 - *Lpadmin (Impr.)*.
 - *nslookup www.google.es*: dice *DNS IP*
 - *wget https://www.000webhost.com/Juan*: también de *sFTP*
 - *SSH*:
 - *CISCO*: de *Config. Switch (NOS)*
 - *ufw*: (Uncomplicated or *Ubuntu Firewall*). *Checksum*.
 - *Nmap*: Escaneo *Puerto* (*Malw. gusano*)

50.5 Facilidades de gestión.

- *Proyec.*: KOrganizer y MSProject
- *Servic. Dir.* (Active Dir.)
- *Kubernetes*
- *Manten.* : parte del *Desarr.*. Mejorar al encontrar *Fall.* (parchear Bugs), *Probl.* y *ReFact..* Resolver Incidencias (*Servic. Dir.*). Ayuda *Preserv. Docum..*
 - Mant. Red: *Instal.* y *Actual.*: a) *Host, Cable o Tecn.*, material fungible b) Licencias, *Probl.* de *Usr, AntiMalw.* y *Ataq.*

Ver *BD Nube, Escal., Monit. (Semaforo)*

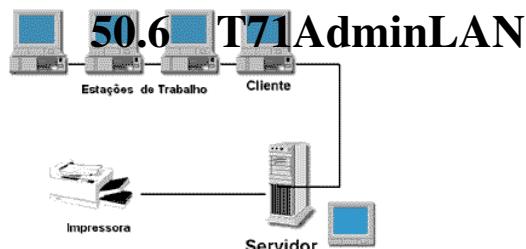
- *Monit.* Red: para que el *Admin.* Red recopil. *Estadísticas* del *Rend.* (uso *Efic.* de *Recursos*) Supervisión, *Anal.*, *Test* o Auditorias en *RT*). Puede recibir *Email* en caso de *Event.* de red (*Congest.* o *Fall.*, permiten aislar y localizar *Probl.*). Tareas:
 - *Sobrecarg. TAP*: de *Servid., MS (Cuota)*.
 - *Segur.*: *Recon.* *Intrusos*, Nombres de *Usr* y *Host* para *Autent., Intercep.* (sniffer) por *Firewall*
 - *Traf.* (sniffer): picos de desvío de Carg., tasas de *Colis.*
 - Otros: *Actual.* y *Manten.* de nuevos *Recursos Compart., BD y Backup*, Licencias y *Actual.*

Algunas *Herram.* o *Apl.* que ayudan son:

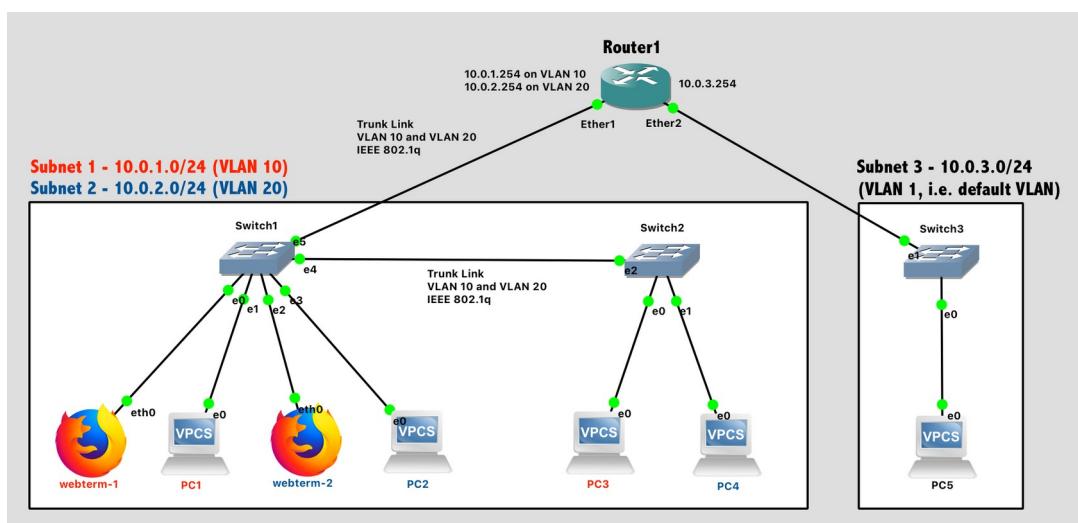
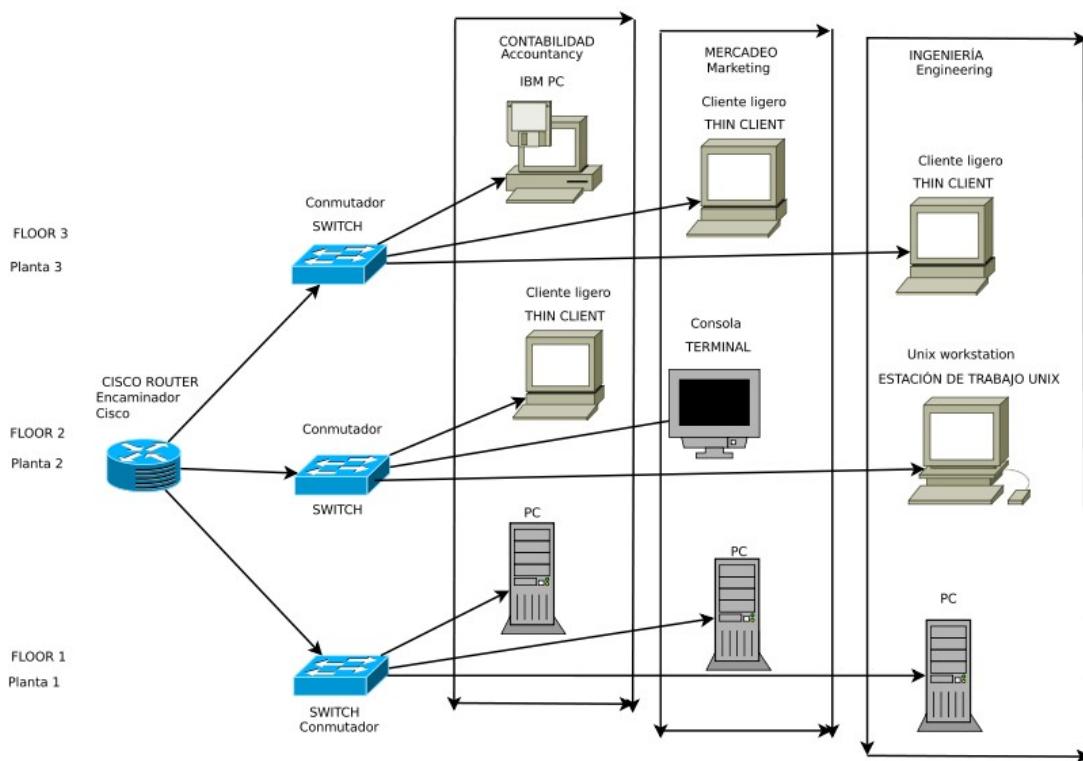
- *logcheck* (*Ubuntu Server, Registr.*). Tener *Disponib.. Control de Firewall* y gest. de *Actual.*
- *Ping* y *Tracer* con ICMP
- **Pandora FMS** (Flexible Monitoring System): *Libre* para M. de cualquier tipo elemento de R.. desde *Servid., Firewall, Proxy, BD, Servid. Web o Rut.* hasta *Impr., IoT* o La Bolsa (enviar SMS si baja una *Empresa*). Mide Balance de Carg.
- **SNMP** (Simple Network Management Protocol, *Protoc. Simple de Admin. de red*): *Estand. del IETF* para recopilar y modificar comportamiento de *Dispos.* de red *TCP/IP*. Los *Dispos.* compatibles suelen ser *Rut. Switch Servid. Impr., Hosts.* Tiene 7 PDUs via *UDP*: *GetRequest, SetRequest,.. (Petic.)*. Para C. Apl.. **RMON** (Remote Network *Monit.*): como *SNMP* es del *IETF* pero para *LAN*.

Ver Func. *SO, Excl. Mut., DevOps, Escan. Puertos (Gusano Malw.), M. Concurr. (Semaforo Hoare), Polling (sondeo, Esper. Activ.)*.

- *Manten.: Testear, Docum.*



LAN VIRTUAL EN UN EDIFICIO DE 3 PLANTAS - THREE FLOOR BUILDING VIRTUAL LAN



Chapter 51

+72. La seguridad en sistemas en red: Servicios de seguridad. Técnicas y sistemas de protección. Estándares. (23FebEncarni)

51.1 Introducción

- Ej.-motivador: *Ataq.* de *Actual.* hospital Prat. *Transac. Sucursal* (SET??)
- Proteger: BD, conex. *SSH*, comunic. *Wireless* (más insegura que *Cable*).
- Hist. : *Cript.* Al-Kindi, Cesar (pero con *Internet* y *RSA* gente normal). Gusano Morris (*Malw.*), *Vulner.* WPS (*WAP*) Mitnick (Ing. Social, *Suplant.*), Spyware Bezos, WannaCry
- Futuro: *Cuant.* contra *Cript.* clásica Mas Segur. todo tiende a *Wireless Fake* y Acoso.

51.2 Ciberseguridad

- Ciber*Segur.* o Hacking: conjunto de acciones (o políticas) para Proteger (ver) la *Inform.* o *Recursos* de un *Sist. Red* (*IoT..*) frente a *Vulner.* y *Ataq.* que pueden ser:
 - A. por *Err.*: catástrofe natural, *Permiso-Grup.* y *BackUp*
 - A. intencionados (FIMI):
 -) Fabric. (*Gener.* o *Prod.*).
 -) *Interr.*
 -) Modif.
 -) *Intercep.*

Según Fig. 51.18, vela la **CIDA** o CIA:

- *Confid.*: rel. con *Intercep.* (con *Permisos*)
- *Integr.*: rel. con **Modif.** (*Suplant.* *Arch.*)
- *Disponib.* o *Acces.*: rel. con *Interr.* (*Permisos*)

- Autent.: rel. *Suplant.* (Fabric. o *Prod. Gener.*)

Políticas de S.: Cebolla (ver) 2 clasif.:

- Fisic, Personal, Organización (Perimetral)
- *TAP*

Ver *BD*, *Transac.*, *Trigger*, *TLS*, *Ataq.*, *Backup* (Copia S. *Preserv.*), *Permiso*, *Instal.* S., *Test Caja Blanca*, *Hackathon (Agil.)*. S. *IPv6 (NAT)*

- RESUMEN

- Segur.: Err. o Ataq., CIDA vs FIMI
- F. Unidir.: Primos, SHA y MD5
- Cript.:
 - a) Simetr.: Cesar, Subst. y AES (usado en *TLS* y *WPA*)
 - b) Asimetr.: Cifr. vs Firm. y RSA (usado en *TLS*)
 -) *TLS*: ej. cript. híbrida AES y RSA
 -) *WPA*: ej. cript. simetr. AES-TKIP y *Vulner.11* y 17
- Autent.: (saber, tener, ser) y A2F, Spoofing (*Voz*)
 -) Firm. vs Certif. (*Blockchain* y NFT)
- Ataq. (FIMI):
 - Fabric.: 1º *Suplant.* InYec. SQL (ver T44BD)
 - Interr.: Disponib. DoS
 - Modif.: *Suplant.*
 - Intercep.: leer *Confid.*, *Fibra* o *Wifi Tapping*. MitM
- Malw.
 - Gusano: Morris y Escaneo *Puertos* usa *Overflow*
 - Ransomware: *Cookie Magica HiJack*
- Suplant. Ing. Social Pharming (*DNS*)
- Vulner.: WPA, MD5, *Overflow..*
- Prote. Física
 - Backup...
- Prote. Personal: *Actual.* SO,
- Prote Org: *Intr. Firewall*, *DesMilit. Proxy*, *VPN*, *SSH*
- Ley LOPDGDD18 adaptada de *Europ.* (ARCOS y AEPD), copyright (*Libre*)
- Estand.
- *Fake* y acoso.

51.2.1 CIDA (no definidos en otros sitios)

- *Confid.* : Inform. solo Acces. por un Grup. de Usr. Se logra con *Permisos*, *Cript.*, *Autent.* o *Fich. Ocultos*. Ver *IP*, *Ataq. Ing. Social*.

51.3 Funciones unidireccionales

- **F. Unidir.** : **P. Fact. Primos** a) $f(p, q) = p * q = 697$ difícil; b) $f(17, 41) = y$ fácil. Otras SHA.
- **Primo** s: *Probl. Fact.* o *DesCompo.* en n° P. y *Isomorf.* de *Graf.*: claramente es *NP* y pero posiblemente no sea *NP-Completo* porque no se ha encontrado *Dem.* Ver *RSA, Servic. Dir., ClPr.* Ver *Matem., Aritm.*
- **SHA** -512: usado en (*EtcShadow*), es una *F. Unidir. Hash*, conviene que sea f. discontinua (caos). Permite detectar no *Integr.*. También **MD5**: *PHP* lo usa para crear un nombre raro *Hash* de un *Fich.* subido, en 1996 se descubrió una *Colis.* que lo hacen *Vuln.* y se está perdiendo no se usa.

51.4 Criptografía simétrica y asimétrica

- En **Cript.** (Encryption) o Cifrado: *Cod.* un *Texto Plano* en *Texto Cifrado* mediante un Alg. *F. Unidir.* para *Confid..* Ej: *TLS* combina Cript. A y *Simetr.*
 - Origen: desde siempre pero gracias a *RSA* y uso másivo de *Internet* hoy mucho.
 - Ej: a) *Protoc.* de Cript.: *TLS, IPSec, (HTTPS, SSH)*. b) Problema del Intercambio de Claves: ver *Intercep., TLS* d) Alg. Encript Simétrico y Longitud de Clave (cambia con L. *Moore*): *3DES, AES*

Ver C. *Present., Segur., E2E. ASN.1, Punt. Java*

- Cript. **Simetr.** (o de 1 sola Clave): *Key* compartida por Emisor y Receptor para *Cod.* aunque se combina con *ASimetr.*
 - Cesar y *Subst.*: Al-Kindi en S. IX hizo *Anal. Prob.* de textos *Hist. R.*
 - Esteganografía: técnicas que permiten ocultar mensajes u objetos, dentro de otros, llamados portadores, para que no se perciba su existencia. ej: hoKI laKU coKE moKO esKA tasKI. Heidi Laymard
 - **AES** (Advanced Encryption Standard): *Estand.* del NIST. El más *Popular* de Cript. S. basado en revolver las letras de el mensaje (cifrado por *Bloq.*) (mediante Cuerpos de Galois+Op. *Matr.*), tanto que salvo que conozcas la clave sea indescifrable (no es *Vuln.* a C. *Cuant.* [Derivando]). AES-256 usa claves de 256 bits. Lo hace el *Chip TPM*. Usado en *WPA2* y *TLS*. Otro es *TKIP*. *PGP* (*Email*).
 - Kerberos: *Protoc.* del *MIT* para *Autent.* dos ordenadores entre si en red insegura *Public.*
 - *Cod.* César (romano): *Hist. R.*

Ver *Equil., TLS, ROM*

- Cript. **ASimetr.** (o de Clave *Public.*): se usan dos *Key*: a) R. genera una clave *Public.* ($Pu = p * q$) de su *Priv.* ((p, q)). b) E. usa *Pu* para encrypt. el Texto $T_e = f(T, Pu)$ (*Hash*). c) Aunque alguien *Intercep.* *Pu* y T_e no podrá obtener *T* pues *f* es *F. Unidir.* (necesita la privada (p, q)). d) Sirve para las 3 de *Cript.*: además de para *Confid.* sirve para *Autent.* e *Integr.* Ej: *Certif. Dig.*

- Sus 2 ramas son:
 - a) Cifrado
 - b) *Firm.* Digit.
- **RSA** (Rivest, Shamir y Adleman): Es el más *Popular* de Cript. ASimetr.. Basado en: nº *Primos* (*Actual.* del orden de 10^{300}) + curvas elípticas. Amenazado por Comp. Cuant. [Shor]. RSA demostraron la existencia de *F. Unidir.* iniciando la Cript. ASimetr. Hist. R.. La hace el *Chip TPM PKCS* (Public-Key Cryptography Estand.): conj. de normas veladas por el RSA California. Ej: el PKCS13 es Cript. de **Curva elíptica** y el PKCS PseudoAleat. Number Generation.
Ej: abajo Fig. 51.18, la *Impl.* real usa el **Tma Chino del Resto** para más *Vel.* en (mod pq usando mod p and mod q [RSA]).
Usado en *TLS*

Ver *VPN, E2E, TLS*

Sean $p = 61, q = 53, n = p * q = 3233$ (n es Público y p, q Privados), sacamos $e = 17, d = 2753$ (exponentes Pu y Pr) que cumplen $de = 1 \pmod{(p-1)(q-1)}$ (efectivamente: $2753 \cdot 17 = 1 \pmod{60 \cdot 52}$). El mensaje encriptado es: $c = m^e \pmod{n}$ y el desencriptado $m_2 = c^d \pmod{n}$

51.4.1 Ej. criptografía simétrica: WPA

- **WPA** (*Wifi Prote.cted Acces.*, no confundir con *WAP*): sistema Certif. del IEEE 802 por la InSegur. de Redes Wireless.
 - Usa AES y/o TKIP (*Temp Key Integr.ity Protocol*).
 - **Vulner.11:** Botón **WPS** (*Wifi Protected Setup*): para Config. y Acces. a Wifi sencillamente. En 2011 Stefan Viehböck descubrió V. *WAP* (*Hist. R.*) basada en *Busq. Fuerza Bruta* que ha sido muy explotada en *Homes LAN* Ej: *WPSApp*.
 - **Vulner.17:** Reemplazó al WEP y en 2017 se descubrió V. del WPA2 que dio el WPA3. Otros WPA Empresarial. Filtr. MAC

51.4.2 Ej. criptografia híbrida: TLS

- *Cript.* Híbrida: ventaja de *Simetr.* (rápido) y Asimetr. (seguro). Ej. *TLS* o *PGP* (Pretty Good Privacy) para *Email*
- **TLS** (Transport Layer Security, no confundir con *TTL*): ej. de *Cript.* Híbrida A+Simetr.. Lleva AES+RSA.
 -) RFCs del IETF para C. *Sesion* (aunque también C. *Transp.??*). Antiguamente SSL (de Empresa Netscape, *Hist. R.*).
 -) *Protoc.* E2E, resuelve la *Intercep.* en 3 pasos de *Comunic.:* a) **Handshake** con *Cript.*HTTPS (*Web*), *FTPS* y *VPN*, *VoIP* e *Email*

51.5 Autenticación

- **Autent.** (Logging abajo): *Id.* o *Certif.* correctamente la *Entidad* (*Usr*, *Fich.*, *Docum.*) frente a un *Sist.* (si tiene *Permiso* (no *Intr.uso*))
 - **No Repudio** (No Rechazo o Irrenuncia): incapacidad de rechazar una *Firm.* como falsa (GAN Aprend. Autom.). *Servic.* que proporciona pruebas de la *Integr.* y origen de los datos en una *Comunic.:*
 - a) N. R. en Origen: Emisor no puede negar que recibió el mensaje porque el Recep. tiene pruebas de envío. *OrCo*
 - b) N. R. Dest.: Recept. no puede negar..
 - **Loggin:** viene de *Arch.* *Log* (logfile), *Registr.* lo que ocurre *Journal.* Al *Rest.* *Sesion.*

En C. *Sesion*, para *Registr.* (de ahí *Fich.* *Log*) o tener *Acces.* a los 4 *Recursos* de red (*Servic.* *Dir.*). Suele haber *Jerarq.* de *Acess* (*Grup.* *Usr*). Es un tipo de *Filtr.* y permite la *Confid..* Formas:

- Saber: *Usr+Pw.* Password, Contraseña, *Key* puede caducar, límite de intentos, Ej: *Comando passwd -e jamc3* cambia en siguiente Login)
- Tener: *Certif.*, *Movil*, *Tarj.*, Llave
- Ser: Biometria huella (*Biolog.*), *Recon.* (Voz con **Antispoofing** *Suplant.* de Alex)
- **A2F** (Autentic. de Doble Factor). Ver *EtcShadow*, *Sesion*
- Kerberos: *Cript.* *Simetr.*
- Otros: *Firewall*, Rel. de Confianza (Active Dir. *Servic.* *Dir.*).
- *Ataq.* A.: ver

Ver *Instal.* *Segur.*, *InTranet.* *Control Domin.* (Active Dir. *Servic.* *Dir.*)

- **ACL (Acces. Control List):** *Tabla* o *BD* con *Permisos rwx* para usar *Recursos* de Red. *Id.* por UID, IP, Protoc., Puerto... Ej.: Active Dir. (*Servic.* *Dir.*), *CISCO Rut.* (Control de Paq.), *TCP Wrapper* (*Daemon Unix* rechaza *Petic.* indeseadas)

51.5.1 Certificado vs firma digital

- **Certif. Digit.:** *Fich.* concedido por un tercero (autoridad certif. como AET y FMNT) para *Id.* *Usr.* Gracias a Cript. *ASimetr.* y *E2E* (basado en X.509). Sirve para las 3 de *Segur.* (*Autent.*, *Confid.* e *Integr.*) ver *Firm.* D.. Ej. *DNIElectr.*
 - **Firma Digit.:** es el 2º campo de la *Cript.* *ASimetr.* e *Inv.* del Cifrado.
Para *Autent.* *Docum.* (No Repudio) y comprobar su *Integr.:* ver Fig. 51.18 a)
E. manda un Doc.+Firma(=HashDoc+ClPr) al R. b) R. descompr. la Firma con ClPu del E. recuperando el HashDoc. Además saca un HashDoc2 c) Si HashDoc2=HashDoc el Doc. es Autent. e Integr. Ver *Certif.*
 - No confundir: Certif. (lo concede una autoridad y permite firmar) y Firma (es el código creado al firmar un documento).
 - Smart Contract: *BlockChain* y *NTFs* (Non Fungible Token).

Ver *Intr.*, CCNA (*CISCO Packet Tracer*), *Test Malw.*, *Fake*, C. *Calidad*

51.6 Ataques

- **Ataq.** : acción que *Vulner.* la *Segur.* de forma intencionada (no confundir con *Err.* que es no intencionada). Clasif. FIMI (*Segur.*): ver Fig. 51.18 [Google-Imagenes: attacks interruption] considera flecha *Flujo R a E*:
 - A. *Fisic.* o cambiar *F. Log.*
 - A. DoS: abajo
 - A. *Autent.*: *Busq.* Fuerza Bruta (WPS (WPA)) o *InYec.* *SQL* (Wordpress, ver T44BD)
 - A. *Ing.* Social: ver *Suplant.*
 - A. *Hist. R.*: WannaCry..., 2023 Clinico Barcelona.
 - **Amenaza** (Threat, no confundir con *Hilo*): pueden ser a) Pasivas: no modif. *Est.* del Sist. (*Confid.*) b) Activas: contrario

Ver *Permiso, Malw.*

- **Ataq.** DoS (Denegación de *Servic.* de Apl.=*Inanic.*): A. *Interr.* a *Disponib..* Basado en *Congest. Puertos* con muchas *Petic.* (*Sobrecarg.*). Se hace con bomba *Fork*. Ver *Intercep.*
- Otras:
Gathering (g. information es reunir info con *Ing.* social de la persona a atacar),
Penesting (penetration test, Ataq. *Simul.* para *Certif.* ej. un *Servid. Web*),
Análisis forense (saber como ha ocurrido Ataq.)

51.7 Malware

- **Malw.** : *Softw.* malicioso. Se evitan con **Antimalw.**, *Herram.* *Recon.* *Integr.* (tripwire, *Certif.*) y seg. perimetral.
 - Virus (no confundir con Gusano abajo): programa que se replica con ficheros .exe y daña un *Fich..* Ej: Un *Admin.* se descarga Pelicula.mp4.exe desde el PC del trabajo (no ve el .exe)
 - Gusano (Worm): al igual que el Virus, progr. que se replica y transmite por muchos Comput en *Red*. El virus necesita que el usuario ejecute un .exe para viajar, el gusano lo hace por el mismo. El virus creo no *Congest. red* pero gusano si.
G. Morris88 usó *Vulner. Buffer Overflow* 1º en *Hist. R.* en ARPANET Ej. **Escaneo de Puertos:** encontrar PC con Puerto TCP 1433 (*Servid. BD SQL*) con Comando *Nmap* (*Ifconfig*)
 - Spyware: *Hist. R.*: divorcio de Bezos (Amazon) por *Movil Espia* (ver *TeleTrabajo*). Ver *Cookie*.
 - Ransomware: pagar *Precio*. Ver **Hijacking** (*Cookie Mágica*).
 - Troyano: M. que se presenta al usuario como un programa aparentemente legítimo pero al ejecutarlo, le brinda a un atacante acceso *Remot.* al equipo infectado.
 - Exploit : programa (o sec. *Comandos*) que explota una *Vulner.* o *Fall.* de un *Sist.* para que instalar otro o hacer que se comporte como se desee. *Backdoor* (puerta trasera) método para *Autent.* en un *Sist.* de forma distinta a la habitual.

- Keylogger:
- Phishing/Pharming (*DNS*) ver *Suplant.*
- *SQL InYec.*: aprovecha *Vulner.* (ver T44BD)
- MalApplet ActiveX (*Progr. Compo.*): Aplic. *JavaScript, Java*

51.8 Suplantación e ingeniería social

- *Suplant.* de *Id.* (Spoofing): ver *Autent., NNTP (Fake), Movil (GSM), Fake.*
 - *Ing. Social*: usar habilidades sociales para obtener *Inform. Confid.*. Iniciada con Mitnick *Hist. R.*
 - Phishing: enviar un *Email* con *Suplant.* de una *Sucursal* u Organismo oficial, pidiendo Datos (con *Truco* e *Ing. Social*). Suelen acortar la URL *Web*. Ej. Pharming
 - Pharming (Farm+Phishing): tipo de Phishing falsificar una *Web*, explotando *Vulner.* del *DNS* (envenenamiento *DNS*)

51.9 Vulnerabilidades: resumen de las mencionadas

- *Vulner.* (debilidad): *Err.* de un *Sist.* empleada por *Ataq.* Exploit (*Malw.*). El hacker la dice. Distinguimos:
 - *Buffer Overflow*: (*Null, Gusano Morris Malw.*)
Cl. Priv. o Publ. (Javier)
MD5 (*Cript.*),
WPA11 (Botón WPS) y 17 (ver).
Pharming (*Suplant. DNS*),
SQL InYec. (ver T44BD)

Ver *WPA2, Exploit (Malw.), Segur., AES, FN, Ataq., Fall., Calidad.*

51.10 Protección: cebolla

- **Prote.** Políticas de *Segur.*: como una Cebolla (onion) **proteger con Capas** la *Inform.* (como un *Home hogar con I/O*):
 - *Fisic.*: llaves en *Centr. Dat.*
 - Personal: sentido común antiphishing
 - Organización o perimetrales:

También clasif. como:

- *Transm.*: En*Cript.*
- *Hardw.*: *Firewall*
- *Softw.*: (*Actual. SO, AntiMalw., Maq. Virt.*)

- Fisicas:
 - 4 de *Instal. Segur.* (*Temp., Tierra...*)
 - *Backup* (RAID, increm. difer.)
 - *Almac. Distrib.* NAS o SAM
- Usuario: muchos ocurren en *SO, Protoc.*, Apl. Red y en *Id. Personal* Ej: *TLS*
 - *EnCRIPT.* (*Protoc. TLS*),
 - Filtr. *Paq.*: *Firewall*, F. *MAC*. Ej: *Admin. Permisos de Rut. Hub.*
 - *Filtr.* Apl. (Alarmas o *Sens.*): *AntiMalw.*, *Email*.
 - *Herram. o Certif.-Test Integr.* Softw. Ej: *Android* (AppStore, Tripware). *SHA The Pirate Bay*
 - IDS (*Intrusiones*)
 - Intermediarios: *Maq. Virt., Proxy, VPN, Exten.* espacio (*InTranet*)
 - Tapar *Vulner.*: *Actual. SO* y *Firmw., Instal. S.* (ver)
 - Sentido Común (*Meta*)
- Organización: las perimetrales

51.11 Seguridad perimetral: interceptación

- *Intercep.* tación (Sniffing): tipo de *Ataq.*
 - *Ataq.* de intermediario o MitM (Man-in-the-Middle)
 - Problema del intercambio de Claves: Sol *TLS* Cript. A+Simetr. o *Cuant.* o *Fibra* (Tapping) o *Wireless* (*Wifi, WPA*)

Ver *Monit.*

51.12 Seguridad perimetral: intrusiones

- *Intr.* usiones (no confundir con *InTranet*): Filtr. y *Recon.* Usr. ajenos (no *Autent.*, *Certif.*). Va en el *Firewall*. Hay varios tipos: HIDS (*Host*) y NIDS (*Red*). Se debe *Admin.* y *Monit. Traf.*.
 - IDS (Intrusion Detection System): no *Bloq. Ataq.* reactivamente (solo Recon. I. escuchando *Traf.*). Ej: *Tripwire*
 - IPS (Intrusion Prevention System): Bloq. Ataq. proactiv. (además de Recon. patrones de comportamiento y lanzar *Event.* Alarma, descarta Paq. y cierra conex.) Ej: *Security Onion*

Ver *Buffer Overflow.*

51.13 Seguridad perimetral: firewall y proxy

- **Firewall**: centraliza el *Traf.* entre red *Priv.* y *Public.* (*PDN*). *Filtr.* la *IP* y los *Puertos* de origen-destino normalmente de *Segment.* *TCP* y *UDP* por *Hardw.* o *Softw.*. *Filtr.* *Acces.* de *Intr.* (forma de *Autent.*) Se pueden *Monit.* *Niveles*:

- F. de PC o personal: *Instal.* en propio PC
- F. *Rut.*: *Filtr.* *Segment.* de *Apl.* e *IPs*
- Dual-*Homed Host* (*NIC*)
- Screened *Subnet*: ver Zona Des*Milit.* (DMZ)

Ej: *Comando Segur.* *Ubuntu ufw (IfConfig)*. Tienen un *NOS*. Ver Escaneo *Puertos* (Gu-sano *Malw.*)

- Zona Des*Milit.* (DMZ): es un *Servid.* con todos los *Puertos* abiertos (Antonio). RE-SUMEN: *Internet* → *DMZ* (\exists *Comunic.* de *Internet* a *DMZ*) pero *Internet* ↔ *RedLocal*, Fig. 51.18.
 - Es similar a una *ExTranet*.
 - Cualquiera Extern. que intente *Acces.* a *RedLocal* no podrá hacerlo (se meterá antes en *DMZ* que es un callejón sin salida).
 - *DMZ*: contiene *Servid.* de *Web*, *SMTP*, *DNS*... emplea *PAT (NAT)*

Ver *ARPA, IP (Intro), Guerra*.

- **Proxy** (RePresentante): *Servid.* intermediario entre *Client.* y *Servid.* *Web*, cualquier *Petic.* hecha por C. pasa por este. Es similar a *MultiTier* en *Servid.* *BD*, añadiendo una *Capa de Segur.* y mejorando el *Rend.* Ej: *Servic. Cache Web* (le quita *Petic.* repetidas al *Serivid.*).
 - Dual/Screened-*Homed*: ver (*NIC o Firewall*)

51.14 Seguridad perimetral: VPN y SSH

- **VPN** (*Virt. Priv. ate Network*): permite comunicar 2 *LAN* mediante *Tunnelling* (ver abajo). Es *Exten.* una *LAN* a *WAN* para conexión *E2E Segur.*
 - Tunelling: *EnCrypt.* y *Encapsul.* los datos que salen de *RedPriv.* sobre una *Red Public.* (*Red Superp.*, *PDN* como *Internet*) usando Cript. *ASimetr.* o *MPLS*. Forma parte de la Suite *Protoc. IPSec*
 - PPTP (Point-to-Point Tunneling Protocol): en C. *Sesion*
 - Ej: conectar *Sucursales* (Fig. 40.10), Trabajo *Remot..* Lleva *TLS*. Ej. de *Protoc. VPN*: OpenVPN, *IPSec* o PPTP (Point to Point Tunnelling Protoc.), *SSH VPN*
- **SSH** (*Secure Shell*): es un *CLI Segur..* Permite *Admin. Remot..* Va en *Puerto 22* y permite *GUI X11*.
 - **Telnet**: menos *Segur.* que *SSH*.

- SSHFS (no confundir con *Almac. Distrib.*): *Sist. Arch. Client.* para *Mount* el Sist. Arch. de un *Servid..* La interacción se hace mediante el Protoc. SFTP que debe tenerlo instalado el Servid. Es un ej. de *VFS* basado en FUSE (Filesystem in Userspace).
- SFTP=SSH+FTP: *Protoc.* creado por el *IETF* para *Exten. FTP* y hacer SSHFS.
- Berkeley r-*Comandos*: reemplazados por SSH (*Hist. R. Hist. SO*). Para *Autent.* en *Unix*, *rsh*, *.rhosts*, *hosts.equiv* confianza transitoria

Ver SSH VPN.

51.15 Estandares: resumen de los mencionados

- Estand. de la Industria:
 - RSA PKCS (arriba)
 - Intel CDSA (Common Data Security Architecture) y *Chip TPM*
 - *TLS*
- ISO/IEC: 27000 para ISM (Information Security Management) para *Segur. Empresa (BD..)*
- IETF: tiene los siguientes grupos de trabajo: Kerberos (*Simetr.*), *IPSec (VPN)*, **X.509**, *DNSSec*, *SSH*, *Certif. Firm. XML*
- NIST: guia SP 800-53
- COBIT (Control Objectives for Information and Related Technology): del ISACA (Information Systems Audit and Control Association) grupo internacional de Auditores
- AET (Agencia Tributaria) y FMNT *Certif. Digit.*
- CCN (Centro Cript. Nacional: ∈ CNI (Centro Nacional de Inteligencia). CCN se divide en CCN-CERT (Computer Emergency Response Team) y ENECSTI (Esquema Nacional de Evaluación y Certificación de la Seguridad de las Tecnologías de Información). Destaca la Serie 800 con políticas y proced. de *Segur.*¹

51.16 Ley LOPDGDD

- LOPDGDD (*Ley* Orgánica de Prote. de Dat. y Garantía de Derechos Digit.): de 2018 adaptada de *Europ.* a España. Derechos del *Usr* y consumidores (de *Empresa*). Ej: Derechos ARCOS. La **Agencia Española de Protección de Datos (AEPD)**: vela que se cumpla la LOPDGDD desde 1992 *Hist. R.*
 - L. *Europ.* : Computer Programs *Directiva 2009* (copyright *Libre*). Ver LOPDGDD (*Ley* ETSI, Proveedor Movil E., Qualcom (*Cuant.*), RAEE (*Ecolog.*)
 - L. de Infraestructuras comunes de telecomunicaciones (ICT): (*Cable* y FTTH)

¹CCN-STIC-800. Las Guías 808 y 807 son las primeras actualizadas a lo dispuesto en el nuevo RD 311/2022.

Además:

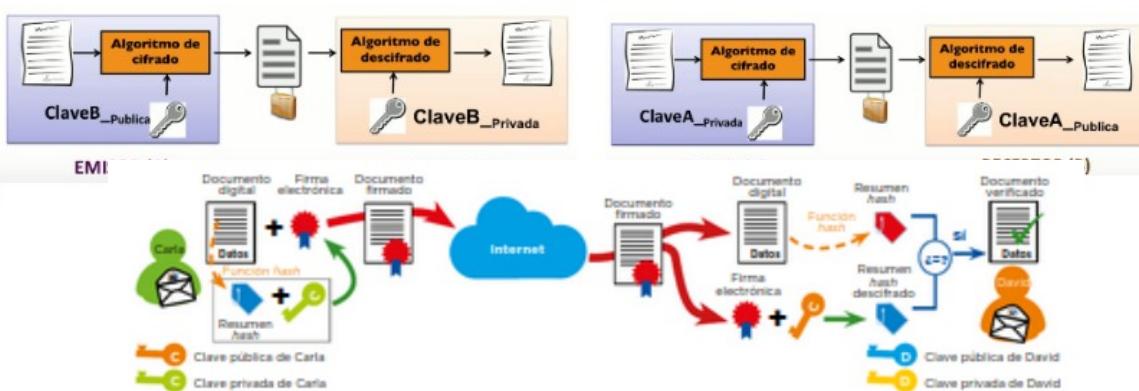
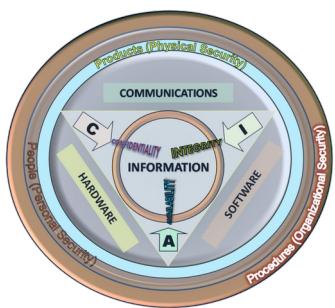
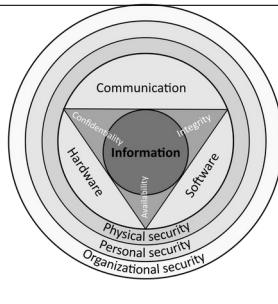
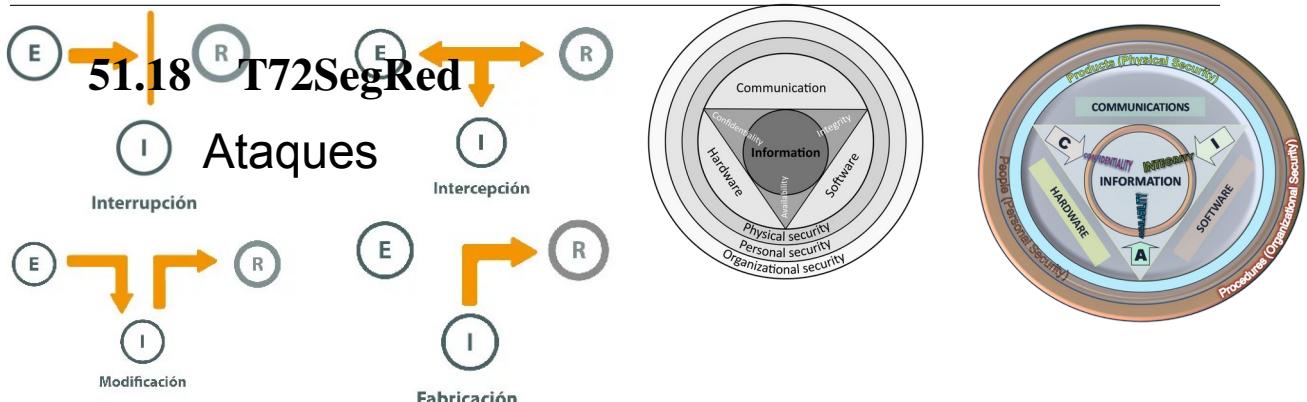
- **Logar.**: L. Mu (*Cuant. Voz*, Crecim. *Fich. (SAM)*)
- *Pot.*: *Seis Grados*
- *Expon.*: *BigO NP*, L. Moore,

Ver RAEE (*Ecolog.*), *Fisic.*, *Boole*, *Conoc.*, *F.*, *Num.*, L. Beer-Lambert (*Aten.*), L. Snell, *Wifi*, *Metr.*, *TV Digit.*

51.17 Fakenews y ciberacoso

- *Fake* News (Bulos, Clickbait): impacto de la *Tecn.*
 - **NNTP** (Network News Transport Protocol): buena IDEA pero se coló la *Suplant.* de Id. en publicar noticias *Fake* IDEA: para evitar *Fake* usar un *Certif.* Trust+ ATP+ NNTP.

Ver *Suplant.*



RSA: ej. Simple de Wiki con los 2 nº (e y d) sacados de CIPr ALICE (recep) BOB (emisor)

$p = 61$	1.er n.º primo privado
$q = 53$	2.º n.º primo privado
$n = p \cdot q = 3233$	producto $p \times q$
$e = 17$	exponente público
$d = 2753$	exponente privado

La clave pública es (e, n) . La clave privada es (d, n) . La función de cifrado es:

$$\text{encrypt}(m) = m^e \pmod{n} = m^{17} \pmod{3233}$$

Donde m es el texto sin cifrar. La función de descifrado es:

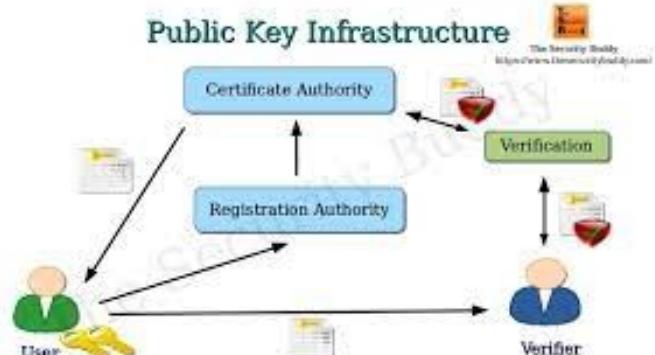
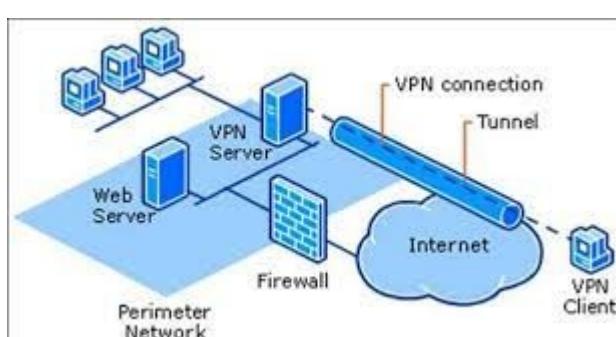
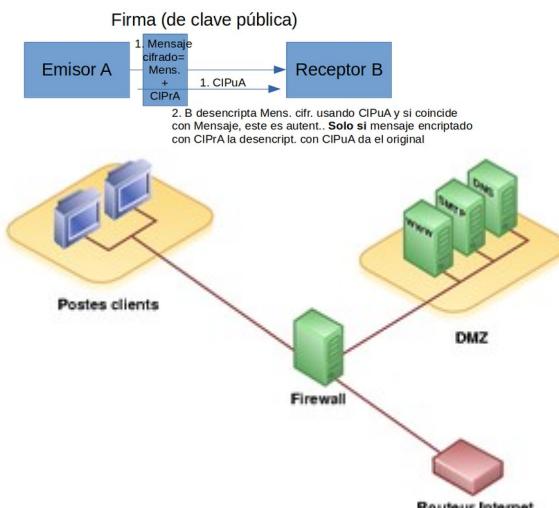
$$\text{decrypt}(c) = c^d \pmod{n} = c^{2753} \pmod{3233}$$

Donde c es el texto cifrado. Para cifrar el valor del texto sin cifrar 123, nosotros calculamos:

$$\text{encrypt}(123) = 123^{17} \pmod{3233} = 855$$

Para descifrar el valor del texto cifrado, nosotros calculamos:

$$\text{decrypt}(855) = 855^{2753} \pmod{3233} = 123$$



Chapter 52

Fórmulas en problemas

52.1 Tablas

- Circunferencia: $P = 2\pi r$ $A = \pi r^2$.
- Compiladores online: https://www.onlinegdb.com/online_c_compiler

Index

- Actuad., 18
Algebr., 8
Amazon, 6
Anal., 8
Andalucia, 5
Antena, 19
Aprend. Autom., 17
Aria, 11
Aritm., 7
ARPA, 5
Array, 26
ATP, 16
Autocompl., 17
Axiom., 15
- B., 5
Bell, 5
BigO, 11
Bioinform., 18
Biolog., 19
Blackbox, 7
Bosque, 17
- Caden., 27
Canon., 12
Card., 11
Cast, 13
Cat., 14
Cient., 12
Clausur., 12
Comple., 15
Compo., 7
Congest., 9
Conj., 11
Consist., 15
Converg., 21
Corr., 9
Correspond., 11
Creat., 16
- Decib., 15
Dem., 15
Dinam., 29
Discrim., 17
- EA, 12
Ecolog., 21
Econom., 20
EM, 19
Emerg., 20
Empresa, 5
Energ., 19
Equiv., 12
Esp., 19
Estat., 26
Etiqu., 17
Evol., 20
- Fake, 21
FBF, 15
Filos., 20
Filtr., 9
Fisic., 18
Flujo, 9
Form., 15
Fract., 8
Frege, 7
Fulkerson, 9
- Geom., 8
Goedel, 15
Google, 6
Graf., 8
- Haz, 14
Hipergraf., 8
Hist. BD, 5
Hist. I., 5
Hist. R., 5
HP, 5
- IA, 17
IBM, 5
- Interf., 7
Interop., 7
Inv., 12
- Jerarq., 8
- Lamb., 13
Lin., 12
Log., 14
Luz, 19
- Markov, 9
Masiv., 16
Matr., 26
Meta, 15
Metr., 10
MIT, 6
Morf., 11
- NLP, 17
NN, 18
Nod., 8
NP, 11
Null, 28
- Omega, 16
Oracle, 5
Ord., 12
Ortog., 8
- Pars., 15
PCA, 9
Perd., 18
Petic., 9
Predic., 14
Prob., 9
Probl., 12
Prop., 15
Pullback, 14
Puls., 8
Punt., 28
- Razon., 16
Recipr., 14
Recon., 17
Reescr., 13
Rel., 11
Robot., 18
- Seis Grados, 10
Semant., 15
Sens., 18
SGD, 17
Signal, 8
Sintax., 15
Sist., 7
Sist. Abierto, 7
SPN, 18
Struct, 27
Subst., 13
- TAI, 15
Tecn., 21
Tip., 13
Tma, 15
Tmp, 19
Topol., 8
Topos, 14
Traduc., 13
Traf., 9
Transduc., 18
Transpar., 7
Truco, 10
Tupla, 11
- U1Arquit, 5
Unif., 6
- VA, 16
Videojuego, 30
Vincul., 13
- Wittg., 20
Wolfram, 19
WYSIWYG, 30