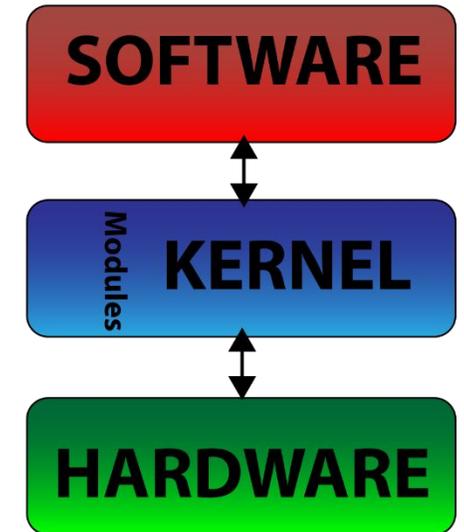
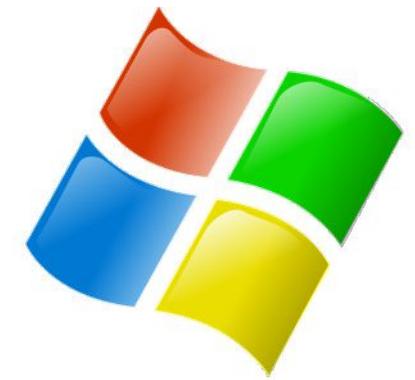




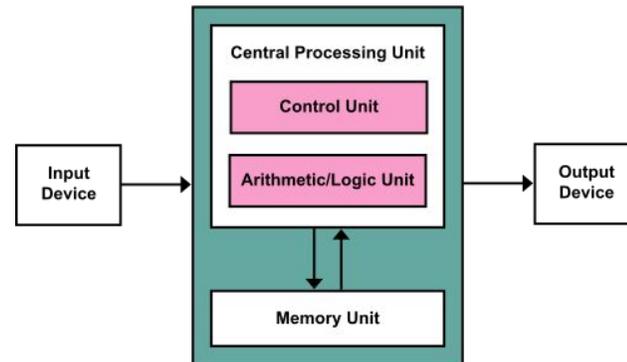
**UNIVERSITÀ DEGLI STUDI
DELLA BASILICATA**

Corso di Sistemi Operativi

File System

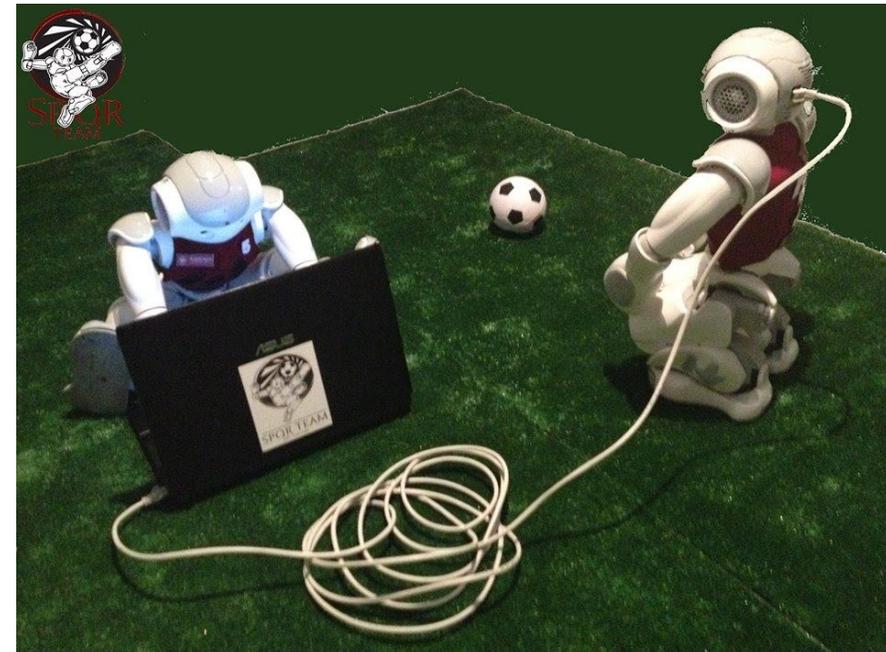
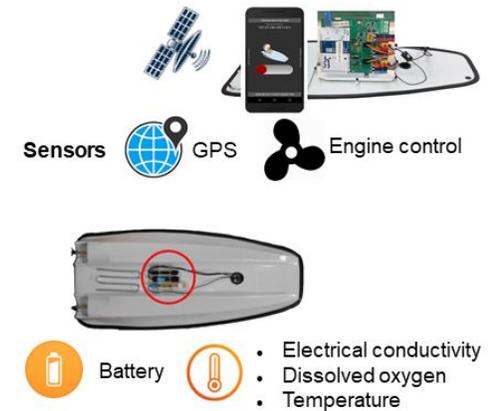


Docente:
Domenico Daniele
Bloisi



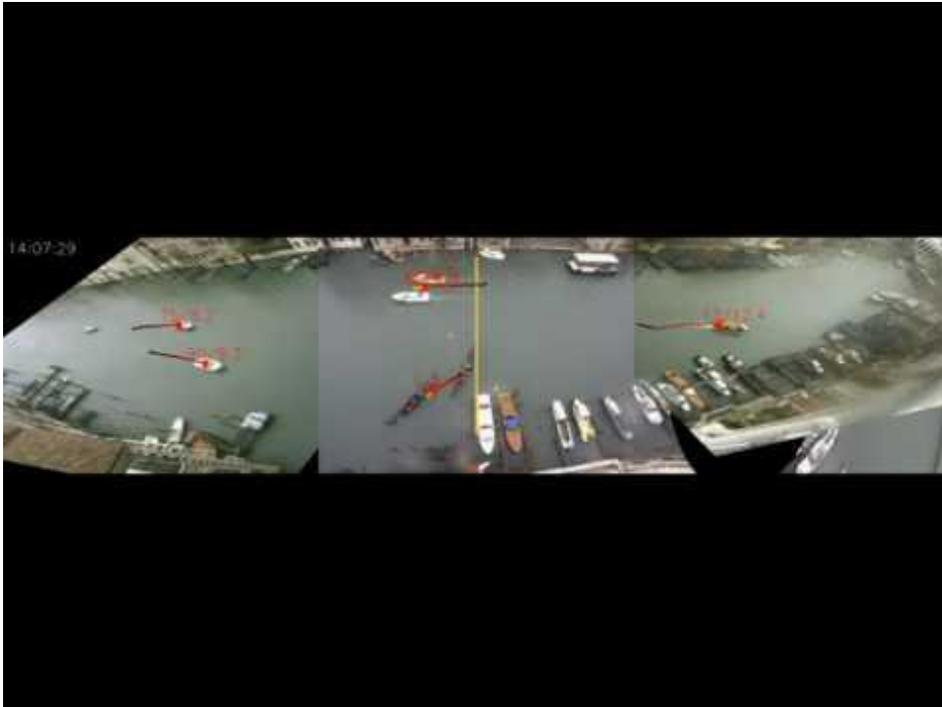
Domenico Daniele Bloisi

- Professore Associato
Dipartimento di Matematica, Informatica
ed Economia
Università degli studi della Basilicata
<http://web.unibas.it/bloisi>
- SPQR Robot Soccer Team
Dipartimento di Informatica, Automatica
e Gestionale Università degli studi di
Roma “La Sapienza”
<http://spqr.diag.uniroma1.it>



Interessi di ricerca

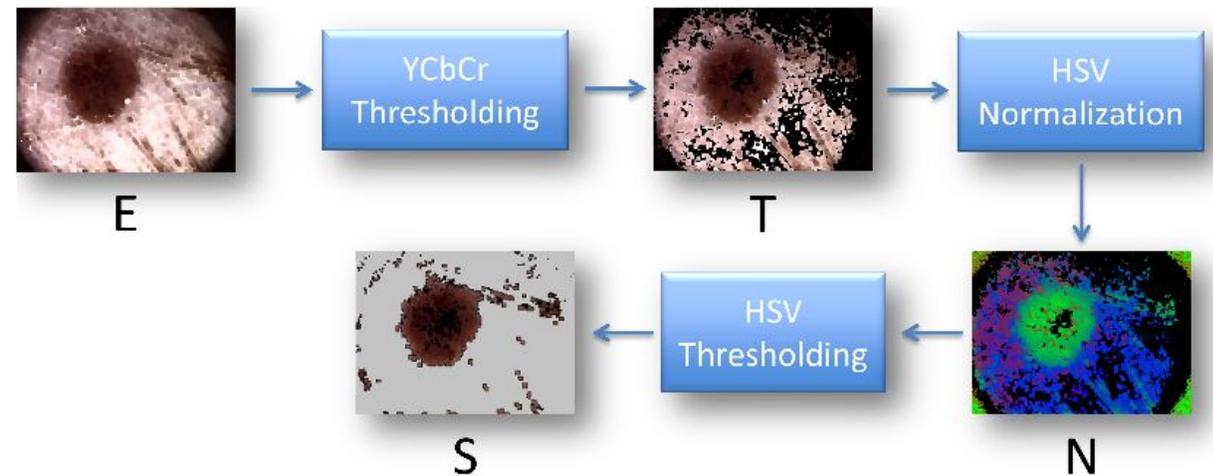
- Intelligent surveillance
- Robot vision
- Medical image analysis



https://youtu.be/9a70Ucgbi_U



<https://youtu.be/2KHNZX7UIWQ>



UNIBAS Wolves <https://sites.google.com/unibas.it/wolves>



- UNIBAS WOLVES is the robot soccer team of the University of Basilicata. Established in 2019, it is focussed on developing software for NAO soccer robots participating in RoboCup competitions.

- UNIBAS WOLVES team is twinned with SPQR Team at Sapienza University of Rome



<https://youtu.be/ji0OmkaWh20>

Informazioni sul corso

- Home page del corso:
<http://web.unibas.it/bloisi/corsi/sistemi-operativi.html>
- Docente: Domenico Daniele Bloisi
- Periodo: I semestre ottobre 2022 – gennaio 2023
 - Lunedì dalle 15:00 alle 17:00 (Aula Leonardo)
 - Martedì dalle 08:30 alle 10:30 (Aula 1)

Ricevimento

- In presenza, durante il periodo delle lezioni:
Lunedì dalle 17:00 alle 18:00
presso Edificio 3D, Il piano, stanza 15
Si invitano gli studenti a controllare regolarmente la bacheca degli avvisi per eventuali variazioni
- Tramite google Meet e al di fuori del periodo delle lezioni:
da concordare con il docente tramite email

Per prenotare un appuntamento inviare
una email a
domenico.bloisi@unibas.it



Programma – Sistemi Operativi

- Introduzione ai sistemi operativi
- Gestione dei processi
- Sincronizzazione dei processi
- Gestione della memoria centrale
- Gestione della memoria di massa
- **File system**
- Sicurezza e protezione

File System

- Il **file system** fornisce il meccanismo per la **memorizzazione in linea** di dati e programmi appartenenti al sistema operativo
- Il **file system** è composto da:
 - un insieme di **file** (contenenti dati)
 - una struttura di **directory** (per organizzare i file)
- Il **file system** risiede, nella maggior parte dei casi, in **memoria secondaria**

File

- Un **file** è un insieme di informazioni correlate, registrate in memoria secondaria, cui è stato assegnato un **nome**.
- Un file **ha attributi** che possono variare secondo il sistema operativo, ma che tipicamente comprendono i seguenti:



Attributi dei file

La Figura 13.1 illustra una **finestra di informazioni** di un file su macOS nella quale sono visualizzati gli **attributi di un file**.

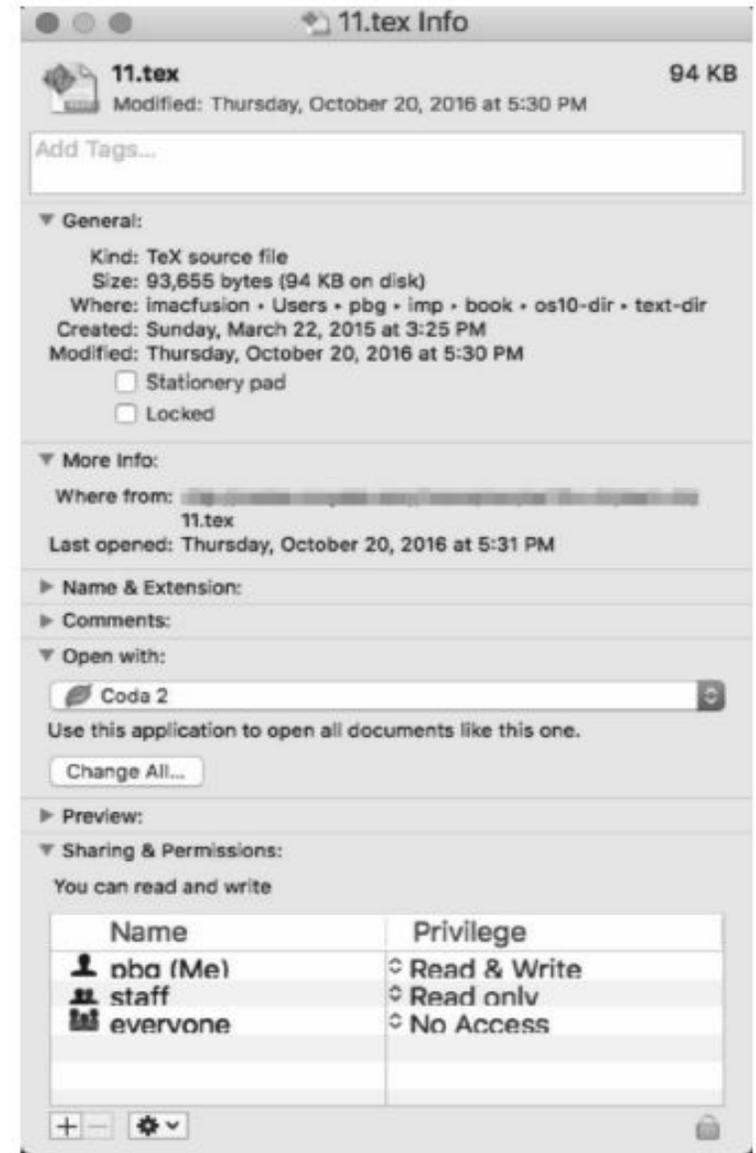
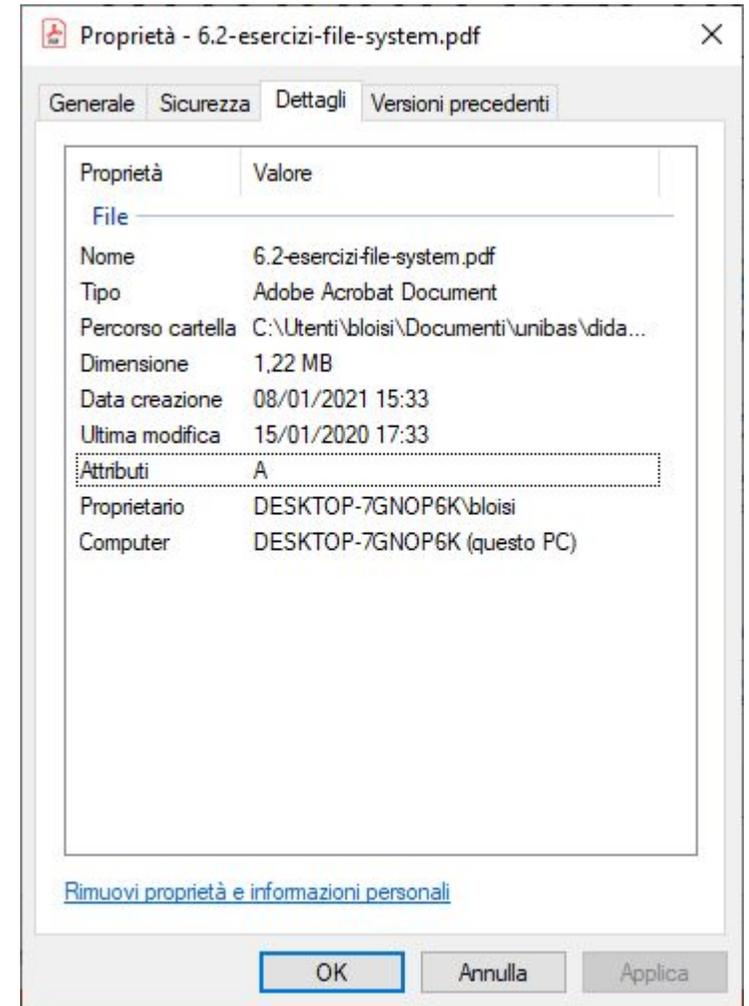
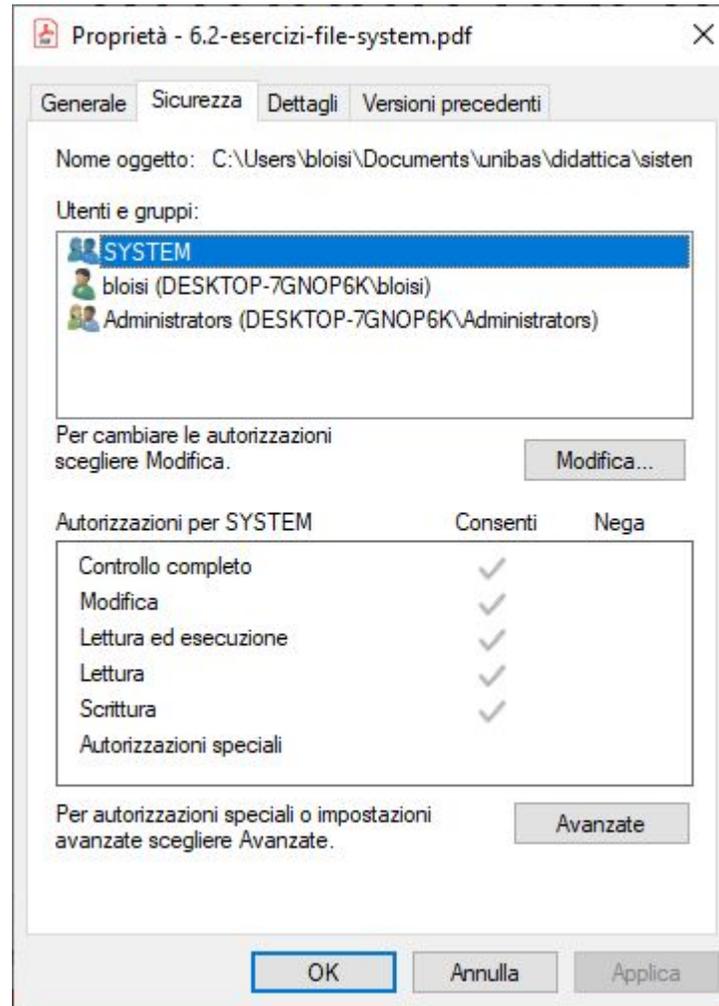
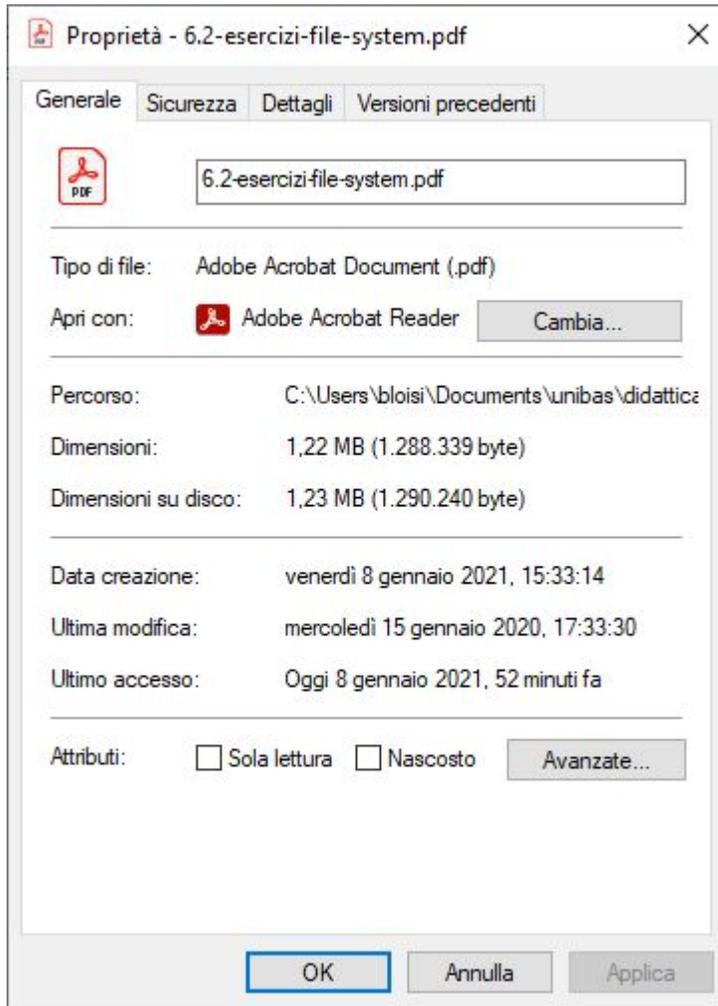


Figura 13.1 La finestra di informazioni di un file su macOS.

Attributi dei file – Windows 10



Operazioni sui file

Creazione
di un file

Scrittura
di un file

Lettura
di un file

Riposizionamento
in un file

Cancellazione
di un file

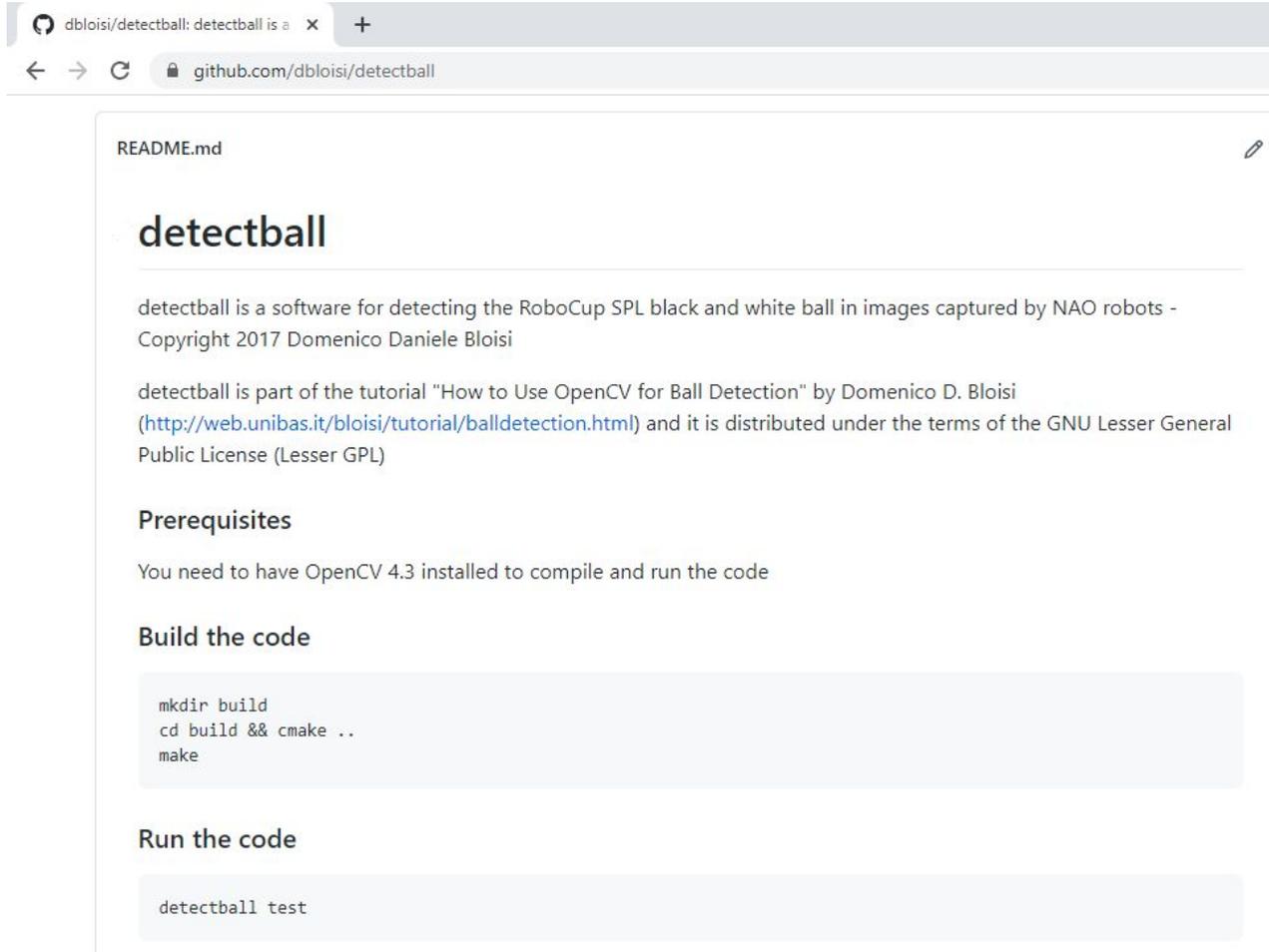
Troncamento
di un file

Tipi di file

Tipo di file	Estensione usuale	Funzione
Eseguibile	exe, com, bin, o nessuna	Programma eseguibile, in linguaggio macchina
Oggetto	obj, o	Compilato, in linguaggio di macchina, non linkato
Codice sorgente	c, cc, java, perl, asm	Codice sorgente in vari linguaggi di programmazione
Batch	bat, sh	Comandi per l'interprete dei comandi
Markup	xml, html, tex	Dati testuali, documenti
Word processor	xml, rtf, docx	Vari formati di word processor
Libreria	lib, a, so, dll	Librerie di procedure per la programmazione
Stampa o visualizzazione	gif, pdf, jpg	File ASCII o binari in formato per la stampa o la visualizzazione
Archivio	rar, zip, tar	File contenenti più file tra loro correlati, talvolta compressi, per archiviazione o memorizzazione
Multimediali	mpeg, mov, mp3, mp4, avi	File binari contenenti informazioni audio o A/V

Figura 13.3 Comuni tipi di file.

Esempio Markdown .md



The screenshot shows the GitHub repository page for 'detectball' by dbloisi. The page is titled 'README.md' and contains the following content:

detectball

detectball is a software for detecting the RoboCup SPL black and white ball in images captured by NAO robots - Copyright 2017 Domenico Daniele Bloisi

detectball is part of the tutorial "How to Use OpenCV for Ball Detection" by Domenico D. Bloisi (<http://web.unibas.it/bloisi/tutorial/balldetection.html>) and it is distributed under the terms of the GNU Lesser General Public License (Lesser GPL)

Prerequisites

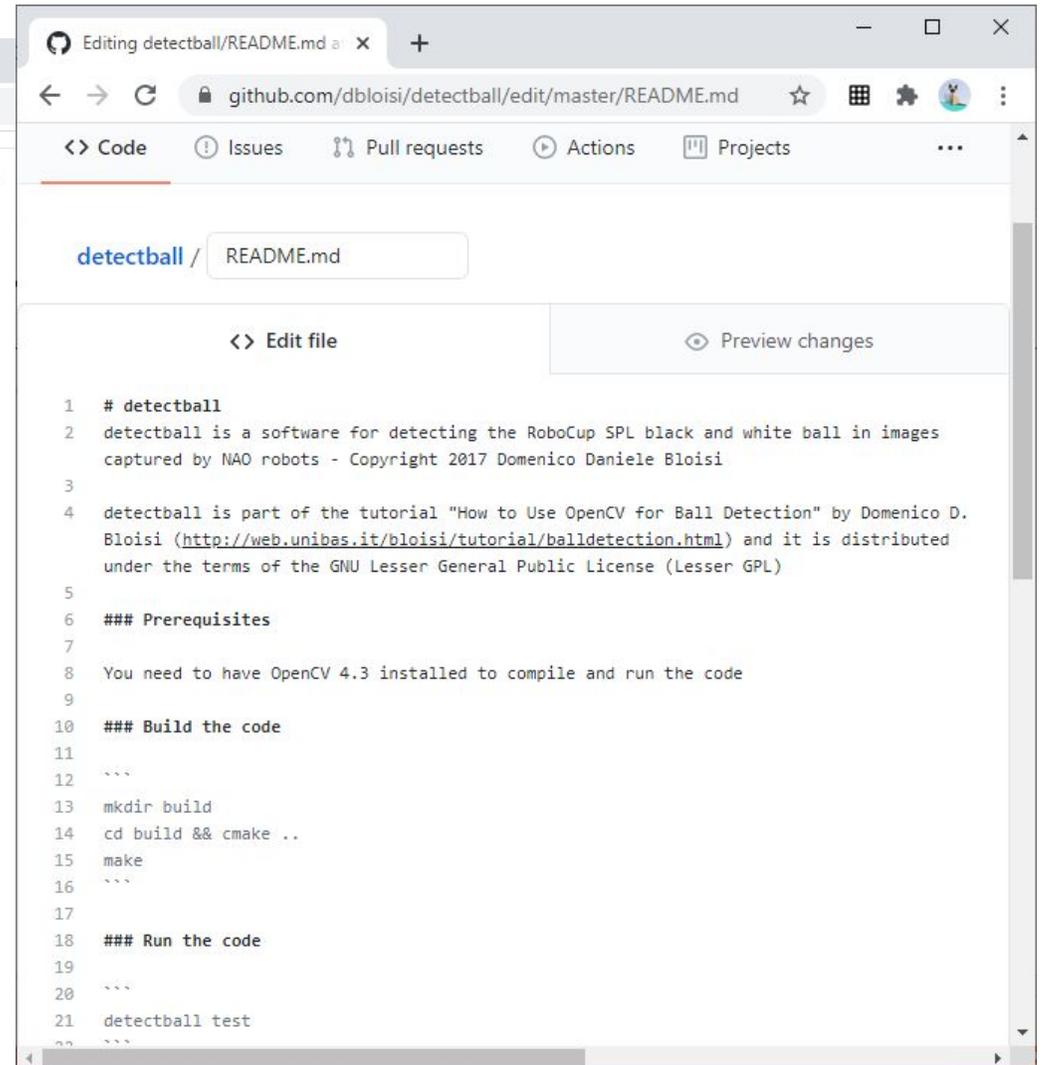
You need to have OpenCV 4.3 installed to compile and run the code

Build the code

```
mkdir build
cd build && cmake ..
make
```

Run the code

```
detectball test
```



The screenshot shows the GitHub editor interface for editing the 'README.md' file. The editor is in 'Edit file' mode and displays the following content:

```
1 # detectball
2 detectball is a software for detecting the RoboCup SPL black and white ball in images
3 captured by NAO robots - Copyright 2017 Domenico Daniele Bloisi
4
5 detectball is part of the tutorial "How to Use OpenCV for Ball Detection" by Domenico D.
6 Bloisi (http://web.unibas.it/bloisi/tutorial/balldetection.html) and it is distributed
7 under the terms of the GNU Lesser General Public License (Lesser GPL)
8
9
10 ### Prerequisites
11
12 You need to have OpenCV 4.3 installed to compile and run the code
13
14
15 ### Build the code
16
17
18
19 mkdir build
20 cd build && cmake ..
21 make
22
23
24
25 ### Run the code
26
27
28
29 detectball test
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

Metodi di accesso

Accesso sequenziale

Accesso diretto

Metodo ad accesso
sequenziale
indicizzato

Accesso sequenziale

Il più semplice metodo d'accesso è l'**accesso sequenziale**: le informazioni del file si elaborano ordinatamente, un record dopo l'altro; questo metodo d'accesso è di gran lunga il più comune, ed è usato, per esempio, dagli editor e dai compilatori.

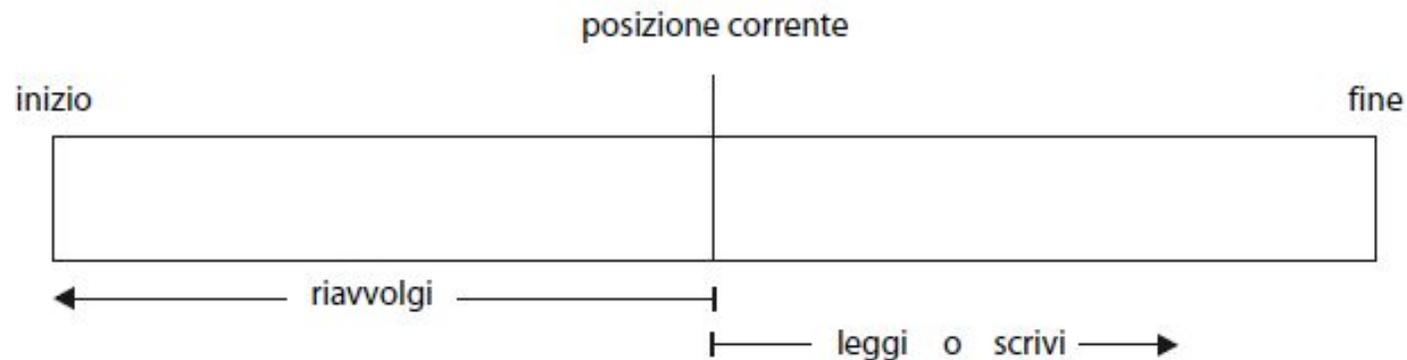


Figura 13.4 File ad accesso sequenziale.

Accesso diretto

- Nel caso dell'**accesso diretto**, un file è formato da elementi logici (**record**) di lunghezza fissa; ciò consente ai programmi di leggere e scrivere rapidamente tali elementi senza un ordine particolare.
- I **file ad accesso diretto** sono molto utili quando è necessario accedere immediatamente a grandi quantità di informazioni (ad es. in un sistema di prenotazione dei voli).

Accesso sequenziale simulato

Non tutti i sistemi operativi gestiscono ambedue i tipi di accesso: Tuttavia si può facilmente *simulare* l'accesso sequenziale a un file ad accesso diretto

Accesso sequenziale	Realizzazione nel caso di accesso diretto
<code>reset</code>	<code>cp = 0;</code>
<code>read_next()</code>	<code>read cp;</code> <code>cp = cp + 1;</code>
<code>write_next()</code>	<code>write cp;</code> <code>cp = cp + 1;</code>

Figura 13.5 Simulazione dell'accesso sequenziale a un file ad accesso diretto.

Accesso sequenziale indicizzato

L'**indice** (*index*) contiene *puntatori* ai vari blocchi; per trovare un elemento del file occorre prima cercare nell'indice, e quindi usare il *puntatore* per accedere direttamente al file e trovare l'elemento desiderato.

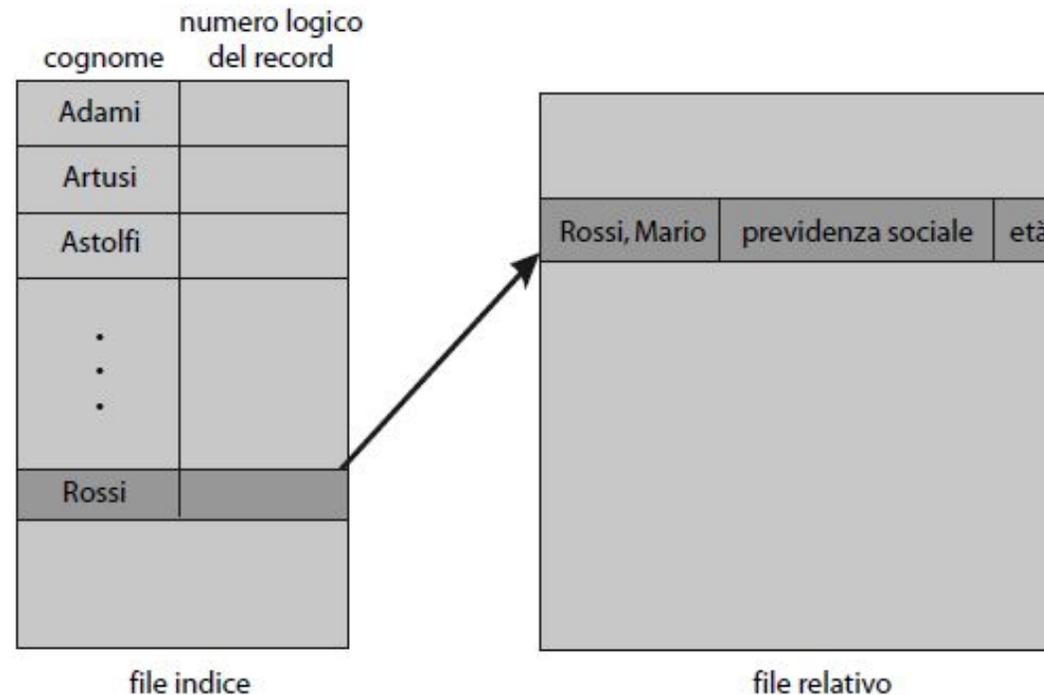


Figura 13.6 Esempio di indice e relativi file.

Directory

La directory è una sorta di **tabella di simboli** che viene impiegata per tradurre i nomi dei file negli elementi in essa contenuti.

```
training_demo/  
├── annotations/  
├── exported-models/  
├── images/  
│   ├── test/  
│   │   ├── 0.png  
│   │   └── train/  
│   │       └── 234.png  
├── models/  
├── pre-trained-models/  
└── README.md
```

Operazioni sulle Directory

Operazioni che si possono eseguire su una **directory**



Directory a un livello

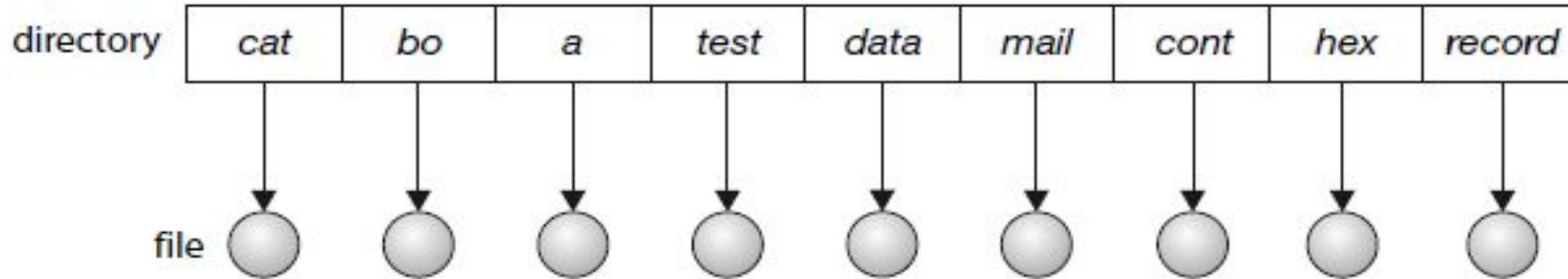


Figura 13.7 Directory a livello singolo.

Directory a due livelli

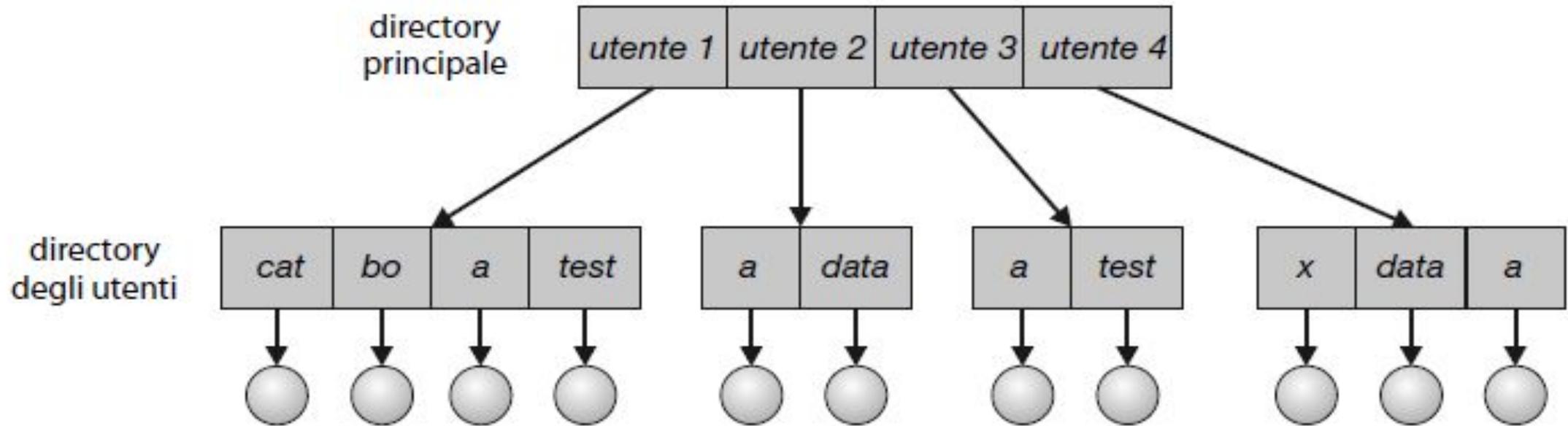


Figura 13.8 Struttura della directory a due livelli.

Directory con struttura ad albero

Questo tipo di struttura permette a un utente di creare sottodirectory in cui organizzare i file

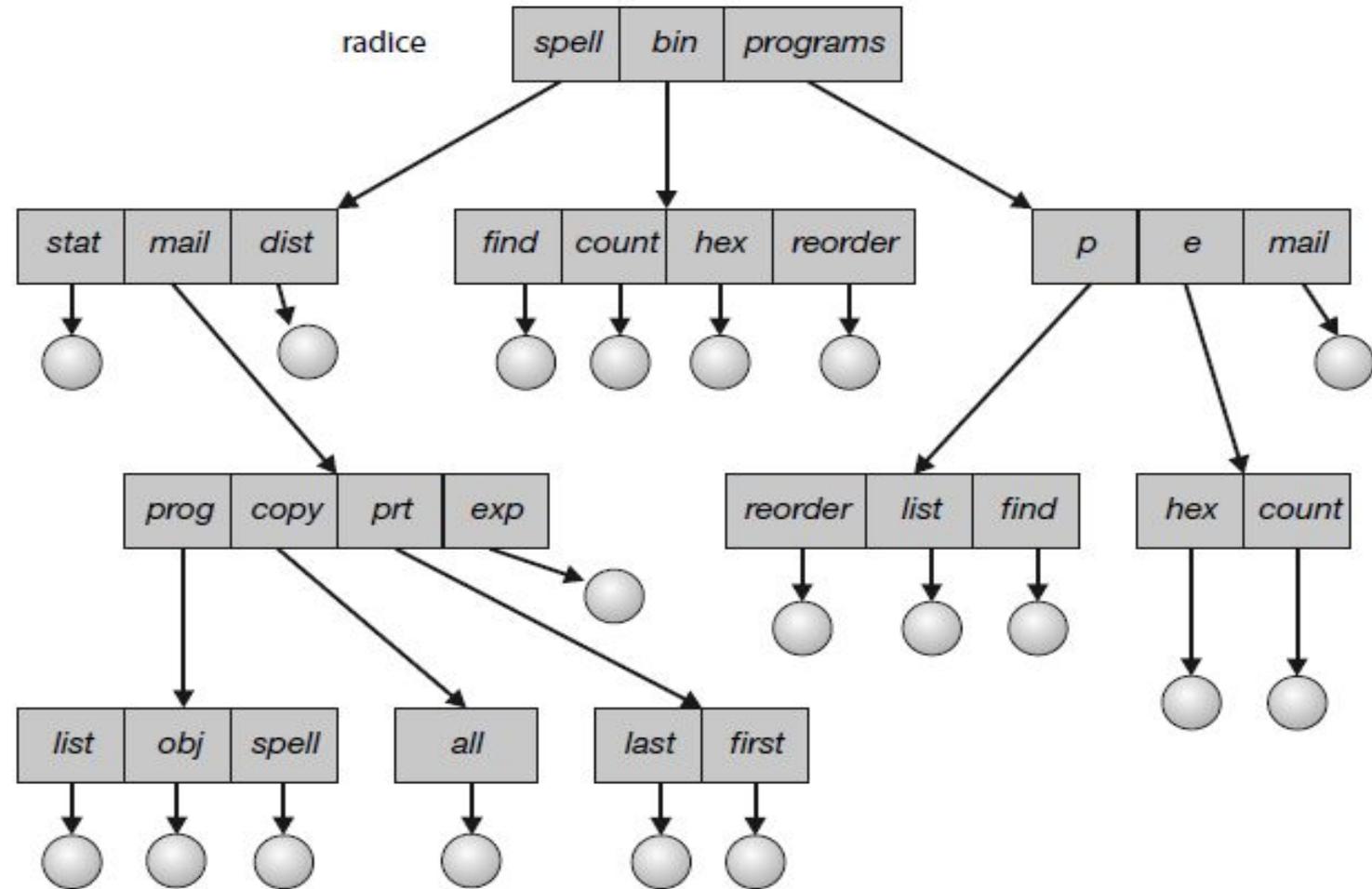


Figura 13.9 Struttura della directory ad albero.

Directory con struttura a grafo aciclico

Le strutture delle **directory a grafo aciclico** permettono la condivisione di sottodirectory e file, ma complicano le funzioni di ricerca e cancellazione.

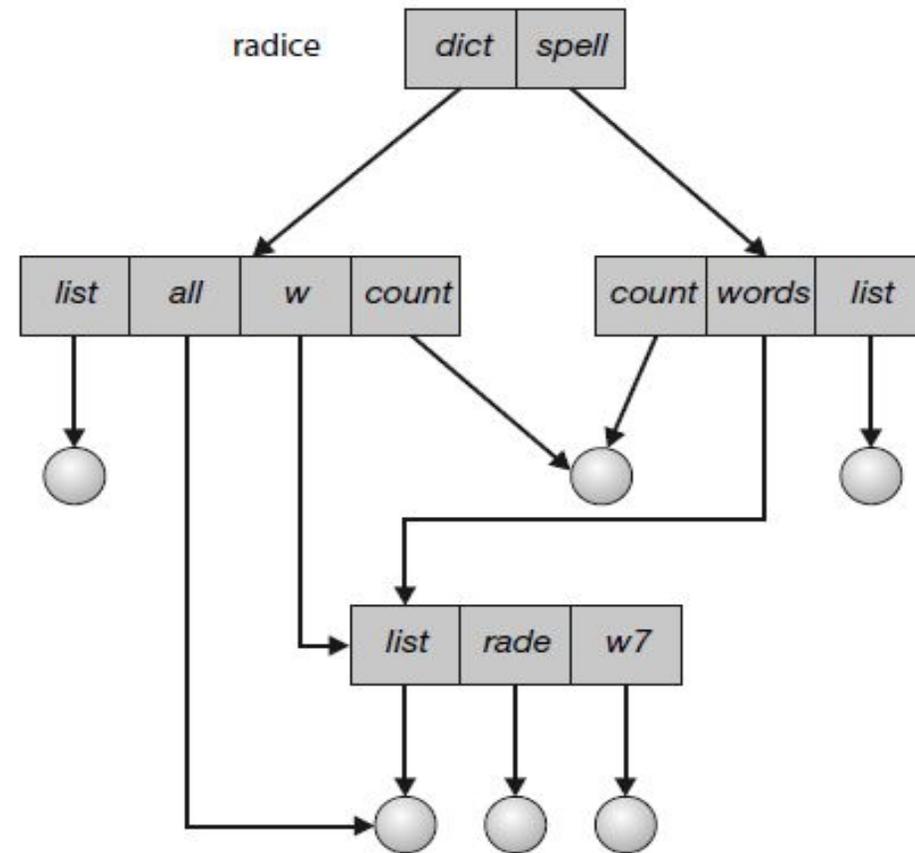


Figura 13.10 Struttura della directory a grafo aciclico.

Directory con struttura a grafo generale

Una **struttura a grafo generale** permette la massima flessibilità nella condivisione dei file e delle directory, ma talvolta richiede operazioni di “ripulitura” (*garbage collection*) per recuperare lo spazio inutilizzato nei dischi

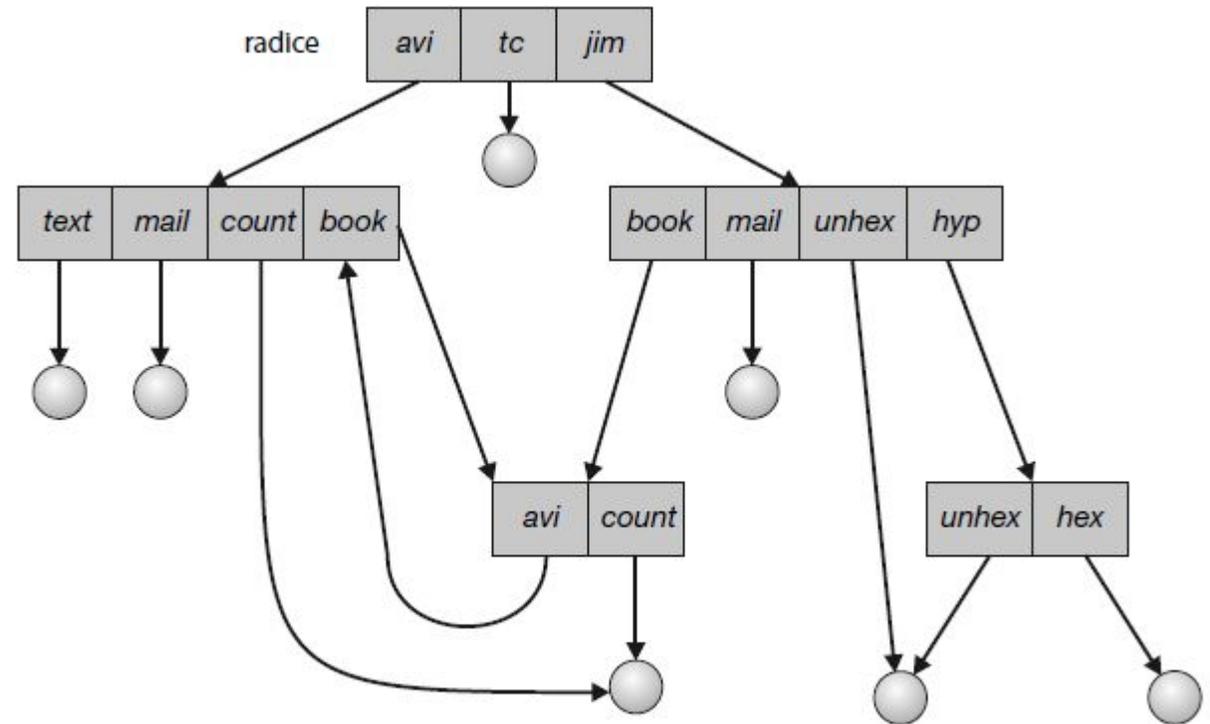


Figura 13.11 Directory a grafo generale.

Protezione

- Le informazioni contenute in un sistema elaborativo devono essere protette dai danni fisici (la questione della *affidabilità*) e da accessi impropri (la questione della *protezione*).
- La necessità di proteggere i file deriva direttamente dalla possibilità di accedervi → **accesso controllato**

Tipi di accesso

Letture

Scrittura

Esecuzione

Aggiunta

Cancellazione

Elencazione

Controllo degli accessi

Nella **lista di controllo degli accessi** (*access-control list, ACL*) sono specificati i **nomi degli utenti** e i **relativi tipi di accesso consentiti**

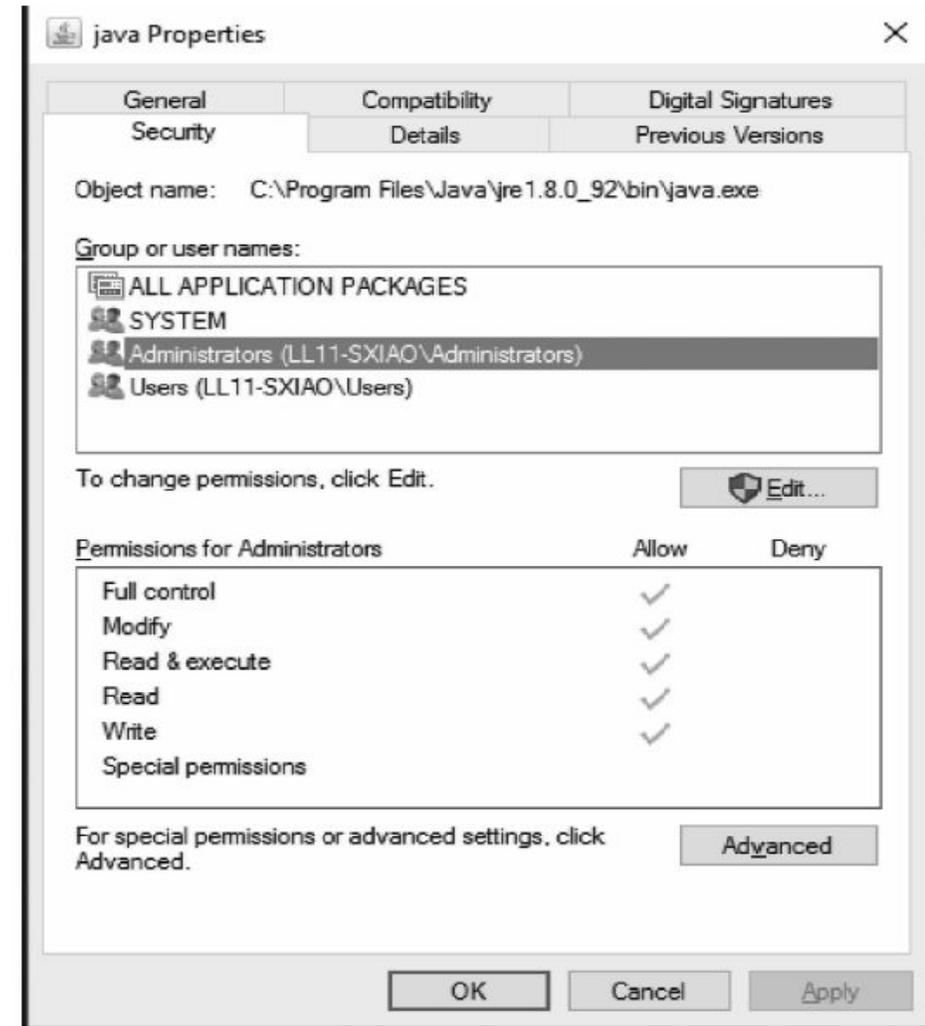


Figura 13.12 Gestione della lista di controllo degli accessi in Windows 10.

File mappati in memoria

In alternativa alla lettura sequenziale di un file sul disco possiamo avvalerci delle **tecniche di memoria virtuale** per trattare l'I/O dei file come l'accesso ordinario alla memoria

↓
mappatura dei file in memoria

↓
una parte dello spazio degli indirizzi virtuali può essere associata logicamente al file

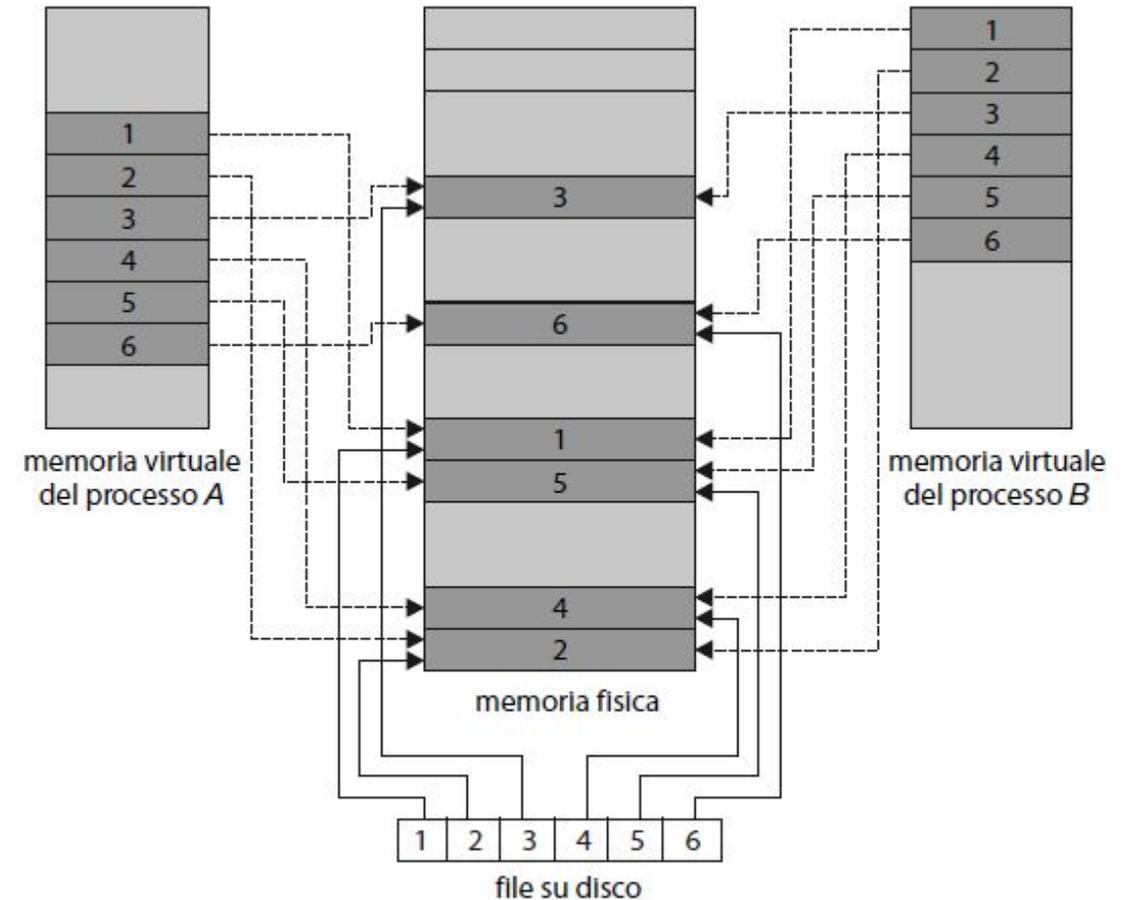


Figura 13.13 File mappati in memoria.

File mappati in memoria

Spesso la memoria condivisa viene implementata utilizzando i **file mappati in memoria**. La **comunicazione fra processi** si ottiene in questi casi mappando in memoria uno stesso file negli spazi degli indirizzi virtuali dei processi coinvolti. Il file mappato in memoria funge da **area di memoria condivisa tra i processi comunicanti** (Figura 13.14)

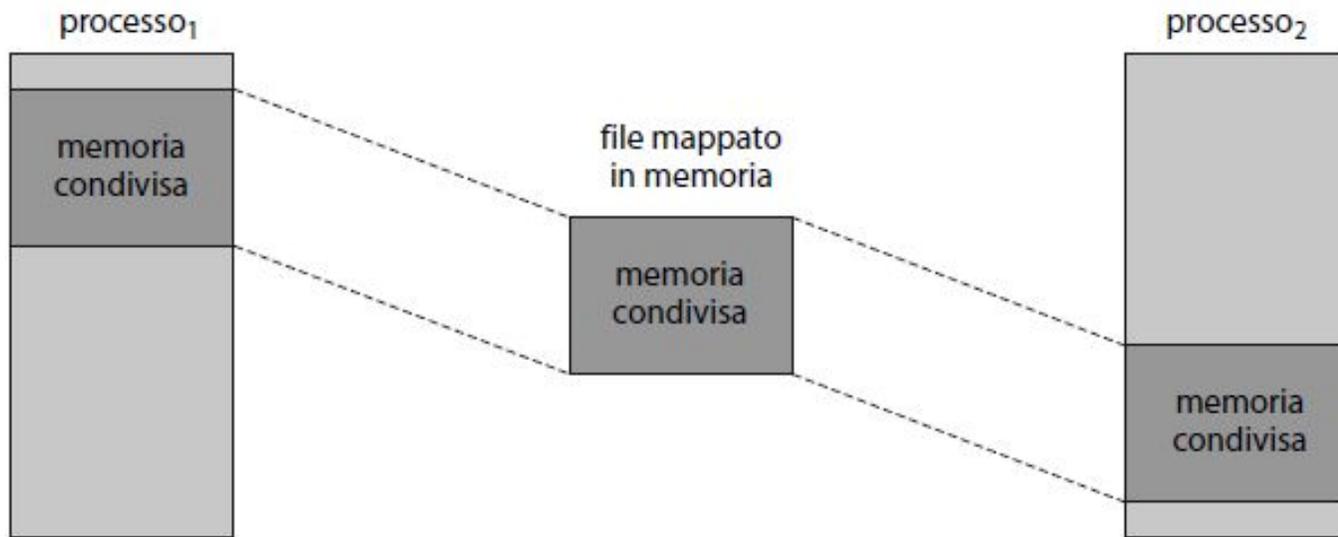


Figura 13.14 Condivisione della memoria tramite I/O mappato in memoria.

Memoria condivisa in Windows

Il **processo produttore** crea dapprima un oggetto di memoria condiviso, sfruttando le funzionalità per la **mappatura di memoria** disponibili nella API Windows. Il produttore scrive quindi un messaggio nella memoria condivisa.

Il **processo consumatore** in seguito (Figura 13.16) crea a sua volta una mappatura del file, e legge il messaggio scritto dal produttore.

Inizio listato Figura 13.15.

```
#include <windows.h>
#include <stdio.h>

int main(int argc, char *argv[])
{

    HANDLE hFile, hMapFile;
    LPVOID lpMapAddress;

    hFile = CreateFile("temp.txt", /* nome del file */
        GENERIC_READ | GENERIC_WRITE, /* accesso R/W */
        0, /* nessuna condivisione del file*/
        NULL, /* sicurezza di default */
        OPEN_ALWAYS, /* apre il file (nuovo o esistente) */
        FILE_ATTRIBUTE_NORMAL, /* attributi del file ordinari */
        NULL); /* niente template del file*/
```

Memoria condivisa in Windows - producer

Completamento listato Figura 13.15.

```
hMapFile = CreateFileMapping(hFile, /* riferimento al file */
    NULL, /* sicurezza di default */
    PAGE_READWRITE, /* accesso R/W alle pagine mappate */
    0, /* mappa l'intero file */
    0,
    TEXT("OggettoCondiviso")); /* oggetto condiviso con nome */

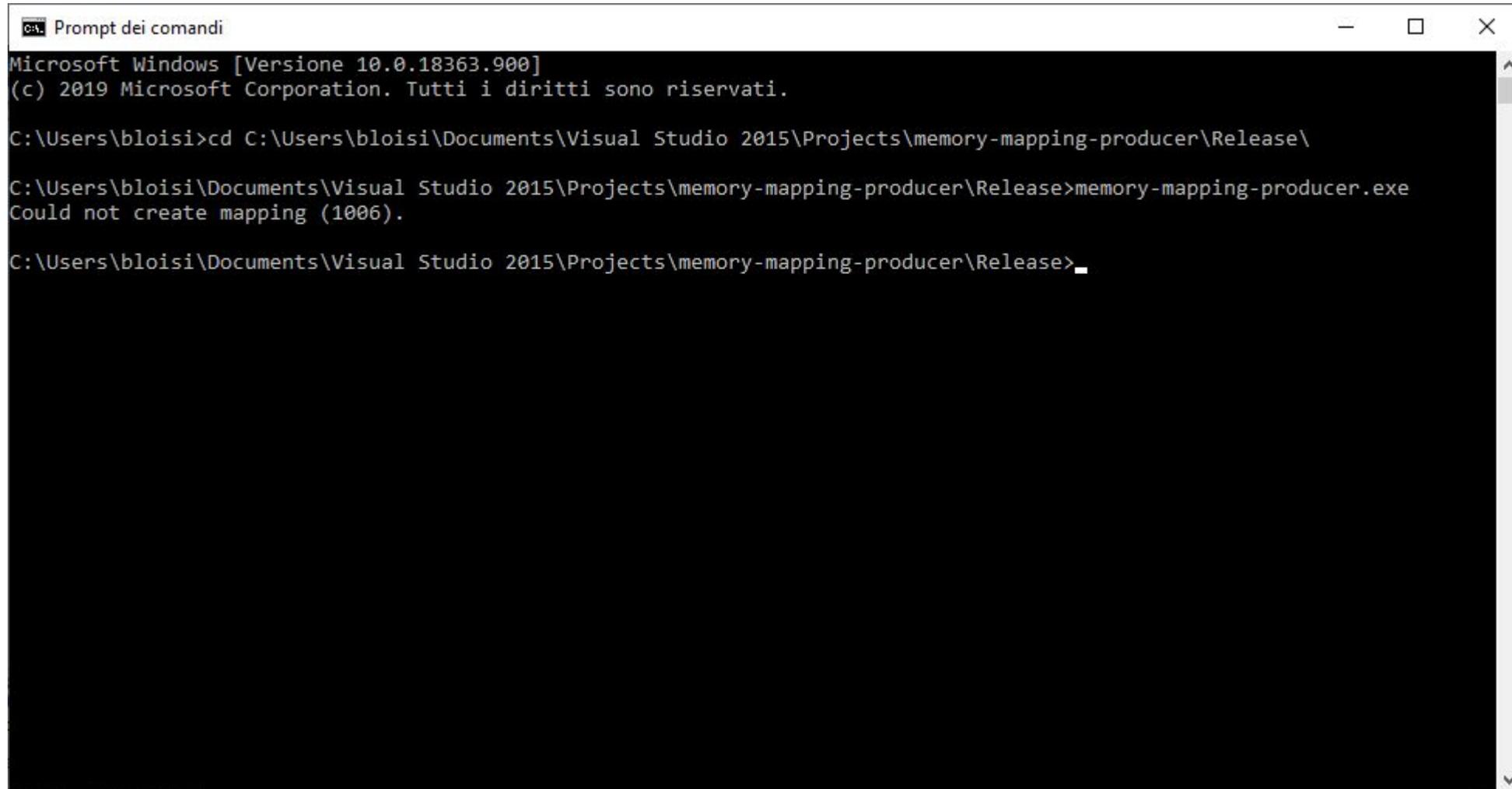
lpMapAddress = MapViewOfFile(hMapFile, /* riferimento al file */
    FILE_MAP_ALL_ACCESS, /* accesso R/W */
    0, /* vista dell'intero file */
    0,
    0);
/* scrive nella memoria condivisa */
sprintf(lpMapAddress, " Messaggio nella memoria condivisa ");

UnmapViewOfFile(lpMapAddress);
CloseHandle(hFile);
CloseHandle(hMapFile);

}
```

Figura 13.15 Produttore che scrive nella memoria condivisa tramite la API Windows.

Memoria condivisa in Windows - producer



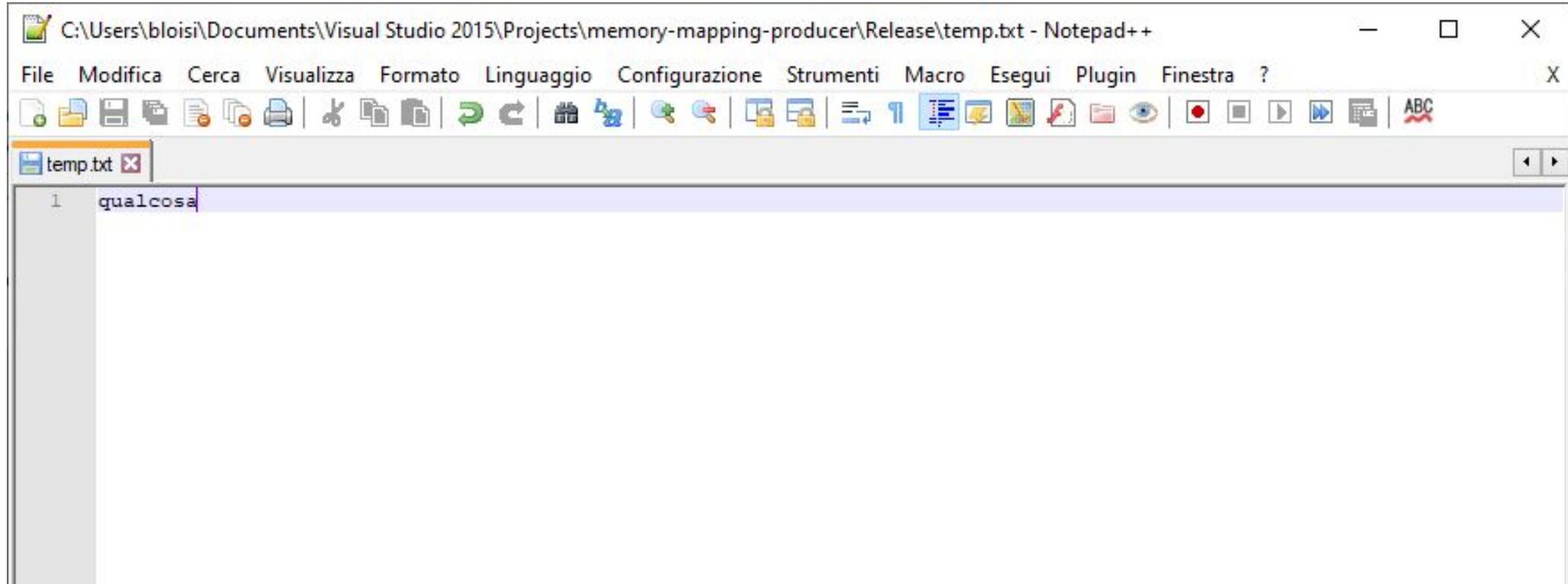
```
ca. Prompt dei comandi
Microsoft Windows [Versione 10.0.18363.900]
(c) 2019 Microsoft Corporation. Tutti i diritti sono riservati.

C:\Users\bloisi>cd C:\Users\bloisi\Documents\Visual Studio 2015\Projects\memory-mapping-producer\Release\

C:\Users\bloisi\Documents\Visual Studio 2015\Projects\memory-mapping-producer\Release>memory-mapping-producer.exe
Could not create mapping (1006).

C:\Users\bloisi\Documents\Visual Studio 2015\Projects\memory-mapping-producer\Release>_
```

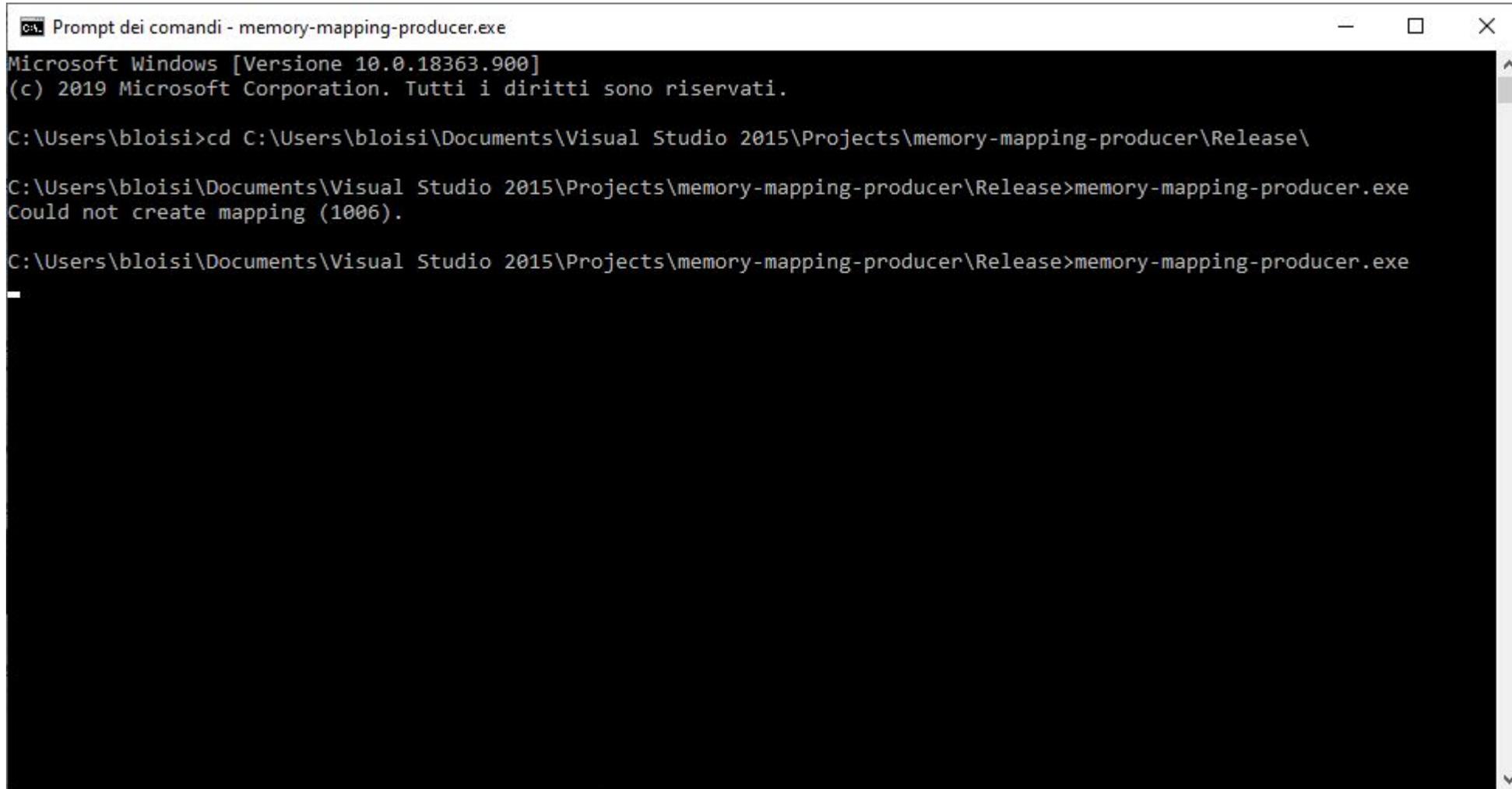
Memoria condivisa in Windows - producer



The image shows a screenshot of a Notepad++ window. The title bar reads "C:\Users\bloisi\Documents\Visual Studio 2015\Projects\memory-mapping-producer\Release\temp.txt - Notepad++". The menu bar includes "File", "Modifica", "Cerca", "Visualizza", "Formato", "Linguaggio", "Configurazione", "Strumenti", "Macro", "Esegui", "Plugin", and "Finestra?". The toolbar contains various icons for file operations and editing. The active tab is "temp.txt". The text area shows a single line of code: "1 qualcosa".

```
1 qualcosa
```

Memoria condivisa in Windows - producer



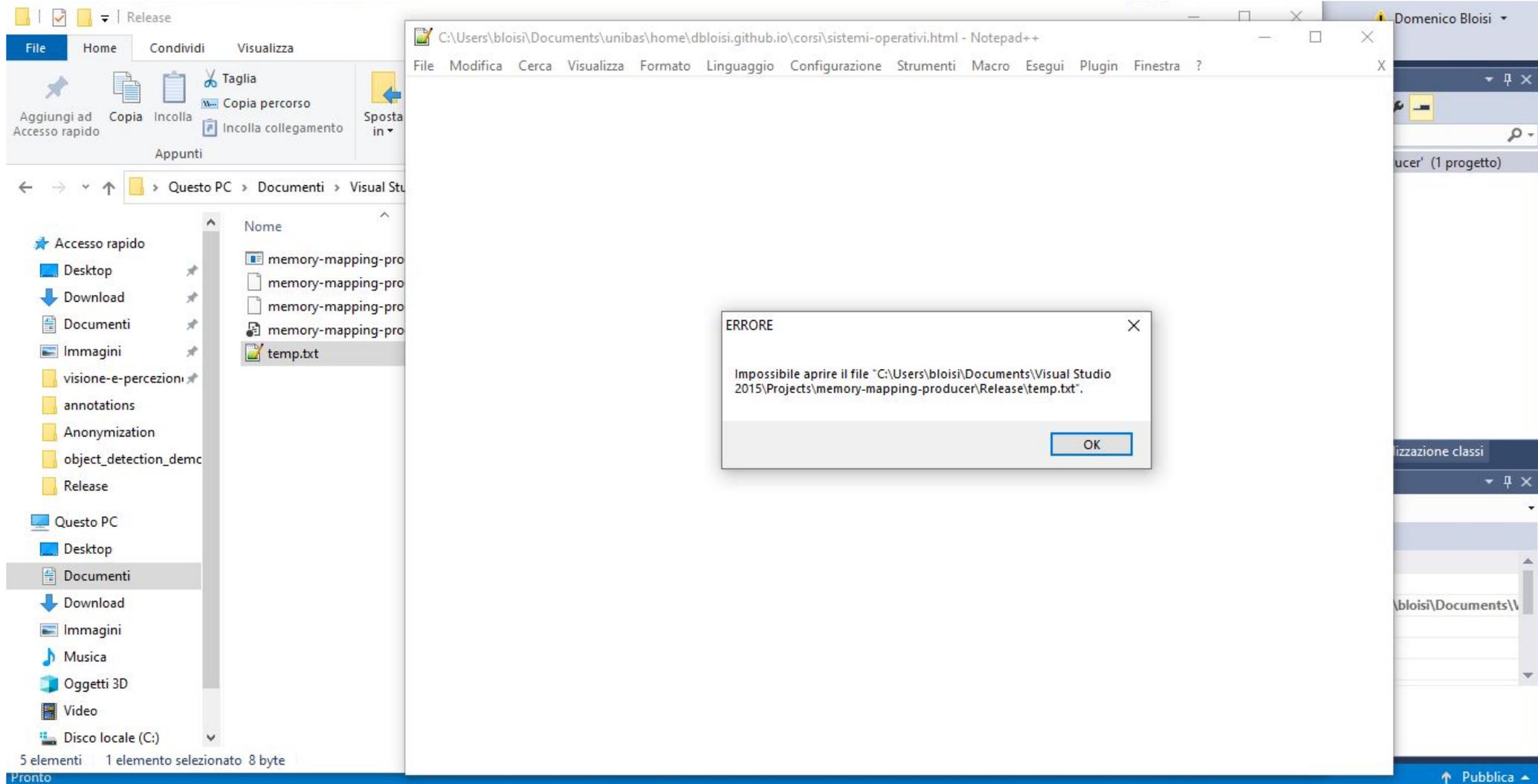
```
Prompt dei comandi - memory-mapping-producer.exe
Microsoft Windows [Versione 10.0.18363.900]
(c) 2019 Microsoft Corporation. Tutti i diritti sono riservati.

C:\Users\bloisi>cd C:\Users\bloisi\Documents\Visual Studio 2015\Projects\memory-mapping-producer\Release\

C:\Users\bloisi\Documents\Visual Studio 2015\Projects\memory-mapping-producer\Release>memory-mapping-producer.exe
Could not create mapping (1006).

C:\Users\bloisi\Documents\Visual Studio 2015\Projects\memory-mapping-producer\Release>memory-mapping-producer.exe
-
```

Memoria condivisa in Windows - producer



Memoria condivisa in Windows - consumer

```
#include <windows.h>
#include <stdio.h>

int main(int argc, char *argv[])
{
    HANDLE hMapFile;
    LPVOID lpMapAddress;

    hMapFile = OpenFileMapping(FILE_MAP_ALL_ACCESS, /* R/W */
        FALSE, /* nessuna ereditarietà */
        TEXT("OggettoCondiviso")); /* nome del file mappato */

    lpMapAddress = MapViewOfFile(hMapFile, /*riferimento al file */
        FILE_MAP_ALL_ACCESS, /* accesso in lettura/scrittura */
        0, /* vista dell'intero file */
        0,
        0);

    /* lettura dalla memoria condivisa */
    printf("Messaggio letto: %s", lpMapAddress);

    UnmapViewOfFile(lpMapAddress);
    CloseHandle(hMapFile);
}
```

Figura 13.16 Consumatore che legge dalla memoria condivisa tramite la API Windows.

Memoria condivisa in Windows - consumer

The screenshot displays the Microsoft Visual Studio IDE with a C program named `consumer.c` open. The code is as follows:

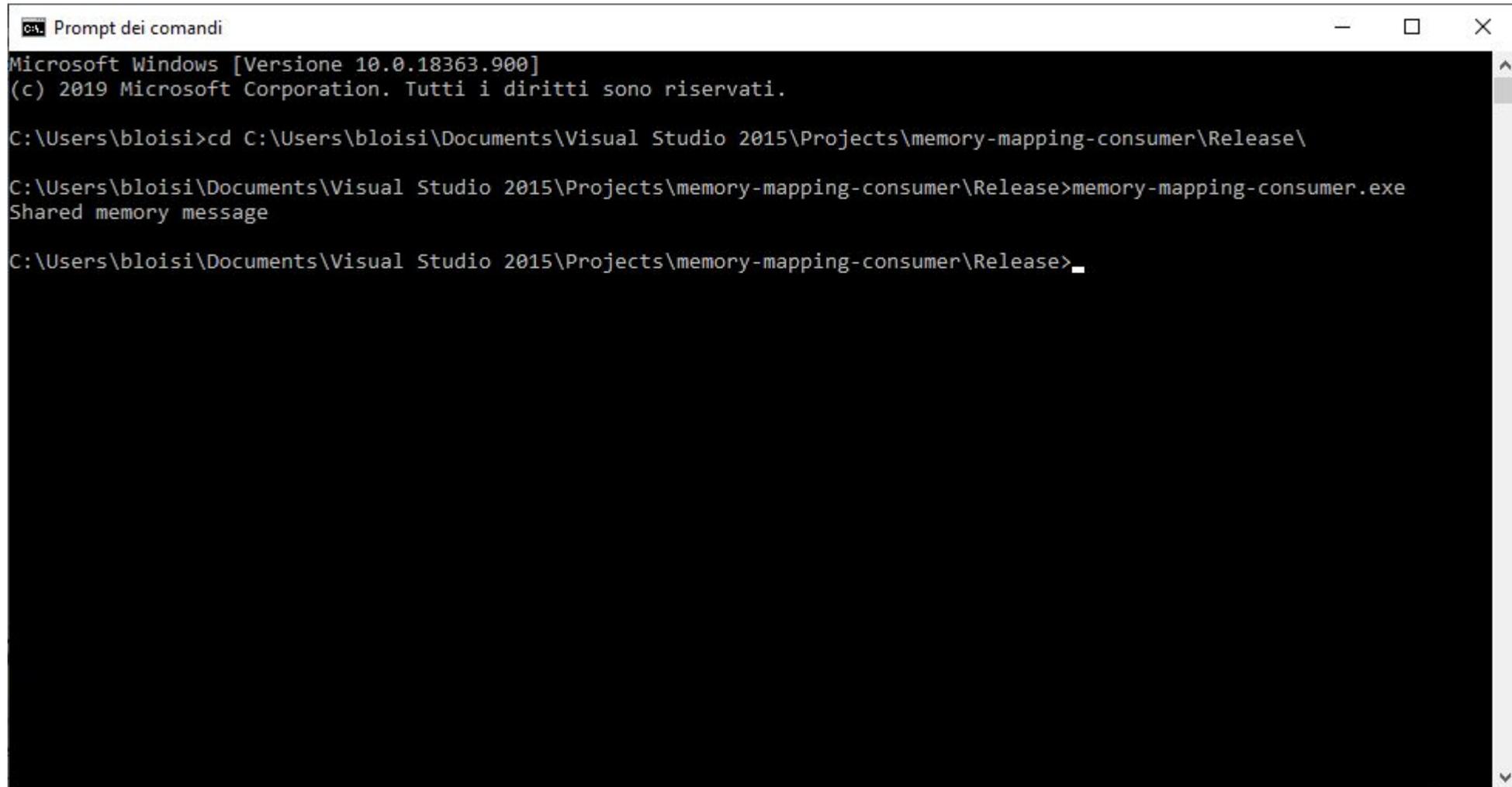
```
2  * Illustrate memory-mapping files in Windows
3  *
4  * Consumer code.
5  *
6  * Figure 13.16
7  *
8  * @author Gagne, Galvin, Silberschatz
9  * Operating System Concepts - Tenth Edition
10 * Copyright John Wiley & Sons - 2018.
11 */
12
13 #include <stdio.h>
14 #include <windows.h>
15
16 int main(int argc, char *argv[]) {
17     HANDLE hMapFile;
18     LPVOID lpMapAddress;
19
20     hMapFile = OpenFileMapping(FILE_MAP_ALL_ACCESS,           // read/write permission
21                             FALSE,                         // Do not inherit the name
22                             TEXT("SharedObject")); // of the mapping object.
23 }
```

The Output window shows the following compilation results:

```
1> 0 functions were new in current compilation
1> 0 functions had inline decision re-evaluated but remain unchanged
1> Generazione codice terminata
1> memory-mapping-consumer.vcxproj -> C:\Users\bloisi\Documents\Visual Studio 2015\Projects\memory-mapping-consumer\Release\memory-map
1> memory-mapping-consumer.vcxproj -> C:\Users\bloisi\Documents\Visual Studio 2015\Projects\memory-mapping-consumer\Release\memory-map
===== Compilazione: 1 completate, 0 non riuscite, 0 aggiornate, 0 ignorate =====
```

The Solution Explorer on the right shows the project structure for `memory-mapping-consumer`, including source files like `consumer.c` and resource files.

Memoria condivisa in Windows - consumer



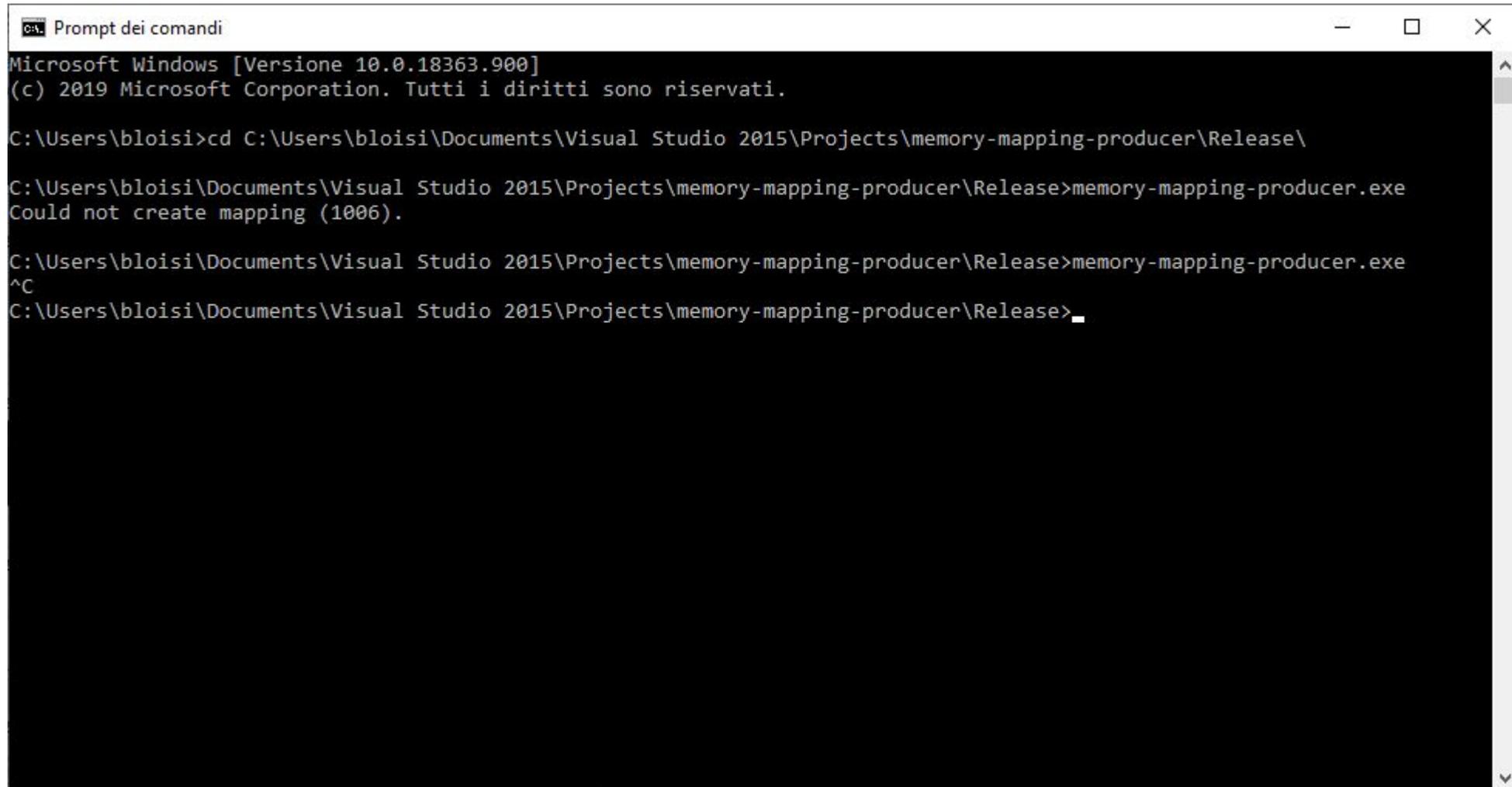
```
Microsoft Windows [Versione 10.0.18363.900]
(c) 2019 Microsoft Corporation. Tutti i diritti sono riservati.

C:\Users\bloisi>cd C:\Users\bloisi\Documents\Visual Studio 2015\Projects\memory-mapping-consumer\Release\

C:\Users\bloisi\Documents\Visual Studio 2015\Projects\memory-mapping-consumer\Release>memory-mapping-consumer.exe
Shared memory message

C:\Users\bloisi\Documents\Visual Studio 2015\Projects\memory-mapping-consumer\Release>
```

Memoria condivisa in Windows - producer



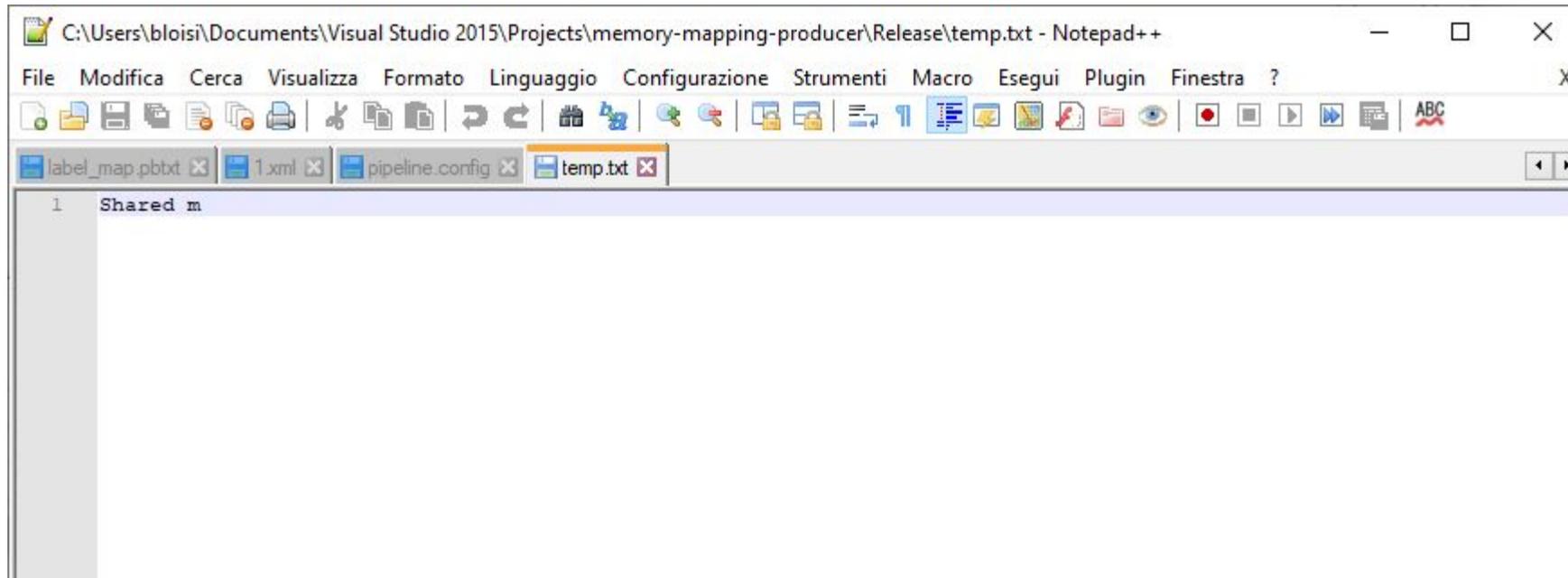
```
Microsoft Windows [Versione 10.0.18363.900]
(c) 2019 Microsoft Corporation. Tutti i diritti sono riservati.

C:\Users\bloisi>cd C:\Users\bloisi\Documents\Visual Studio 2015\Projects\memory-mapping-producer\Release\

C:\Users\bloisi\Documents\Visual Studio 2015\Projects\memory-mapping-producer\Release>memory-mapping-producer.exe
Could not create mapping (1006).

C:\Users\bloisi\Documents\Visual Studio 2015\Projects\memory-mapping-producer\Release>memory-mapping-producer.exe
^C
C:\Users\bloisi\Documents\Visual Studio 2015\Projects\memory-mapping-producer\Release>_
```

Memoria condivisa in Windows - producer



The image shows a Notepad++ window with the following details:

- Window title: C:\Users\bloisi\Documents\Visual Studio 2015\Projects\memory-mapping-producer\Release\temp.txt - Notepad++
- Menu bar: File, Modifica, Cerca, Visualizza, Formato, Linguaggio, Configurazione, Strumenti, Macro, Esegui, Plugin, Finestra, ?
- Toolbar: Standard editing and development tools.
- Tab bar: label_map.pbt.txt, 1.xml, pipeline.config, temp.txt (active)
- Text content: Line 1: Shared m

Realizzazione del file system

- Descriveremo, per semplicità, i file system che risiedono sul più comune tipo di memoria secondaria, cioè il disco.
- Per fornire un efficiente e conveniente accesso al disco, il sistema operativo fa uso di uno o più **file system** che consentono di *memorizzare, individuare e recuperare* facilmente i dati.

Struttura del file system

Per migliorare l'efficienza dell'I/O, i trasferimenti tra memoria centrale e dischi si eseguono per **blocchi**. Ciascun blocco nel disco è composto da uno o più **settori**.

Il **file system** è generalmente composto da più **livelli** distinti

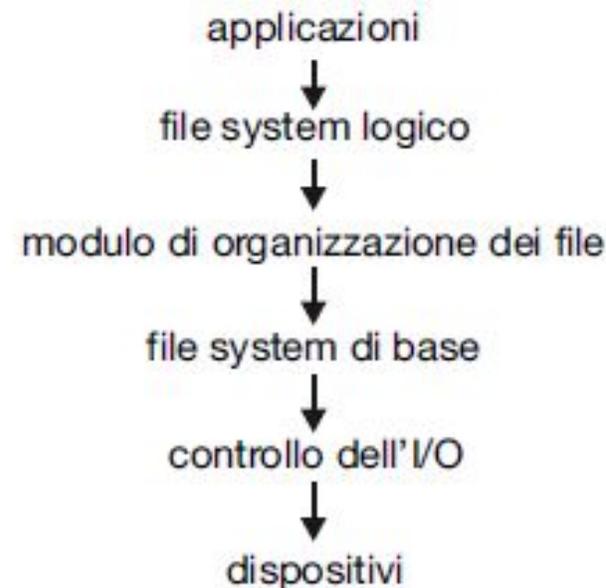


Figura 14.1 File system stratificato.

Operazioni del file system

Per realizzare un **file system** si usano diverse strutture dati, sia nei dischi sia in memoria. Queste strutture variano a seconda del sistema operativo e del file system, ma esistono dei principi generali.

Fra le strutture presenti nei dischi ci sono le seguenti:

blocco di
controllo
dell'avviamento

blocco di
controllo
del volume

struttura
della directory

blocco di
controllo del file
(FCB)

Blocco di controllo del file

Il **blocco di controllo del file (FCB)** contiene molti dettagli del relativo file. Ha un identificatore unico per poterlo associare a una voce della directory.

permessi per il file
data e ora di creazione, di ultimo accesso e di ultima scrittura
proprietario del file, gruppo, ACL
dimensione del file
blocchi di dati del file o puntatori a blocchi di dati del file

Figura 14.2 Tipica struttura di FCB (*file-control block*).

Apertura e lettura di file

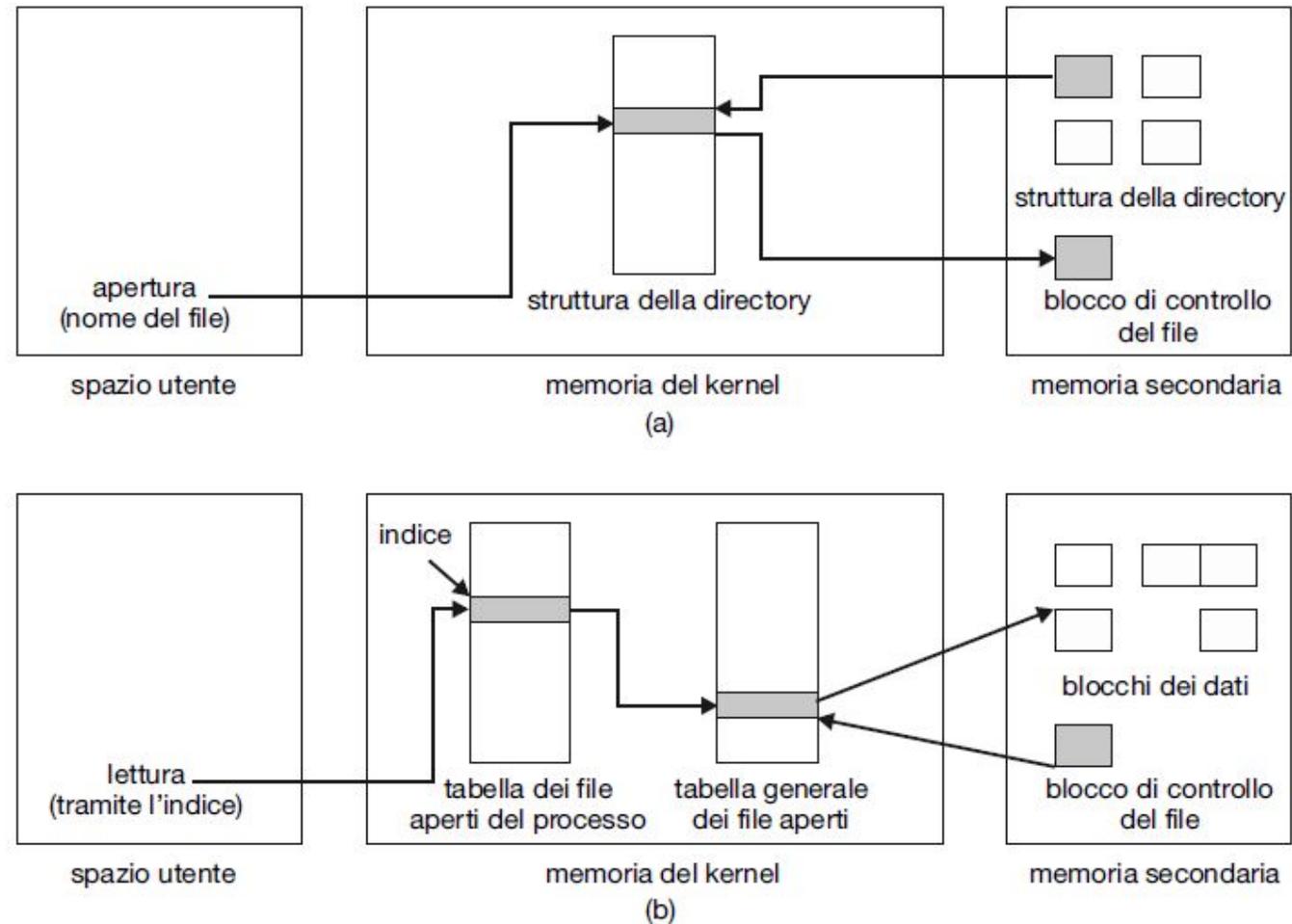


Figura 14.3 Strutture del file system che si mantengono nella memoria; (a) apertura di file; (b) lettura di file.

Realizzazione delle directory

Lista lineare

Tabella hash

Il più semplice metodo di realizzazione di una directory è basato sull'uso di una **lista lineare** contenente i nomi dei file con puntatori ai blocchi di dati.

Questo metodo è di facile programmazione, ma la sua esecuzione è onerosa in termini di tempo.

Un'altra struttura dati che si usa per realizzare le directory è la **tabella hash**, che riceve un valore calcolato a partire dal nome del file e riporta un puntatore al nome del file nella lista lineare. Offre diminuzione del tempo di ricerca nella directory, ma presenta il problema delle **collisioni**.

Metodi di allocazione

Esistono tre metodi principali per l'allocazione dello spazio di un disco:

Allocazione
contigua

Allocazione
concatenata

Allocazione
indicizzata

Allocazione contigua

L'**allocazione contigua** può risentire di **frammentazione esterna**. Lo spazio contiguo si può allargare attraverso delle estensioni allo scopo di aumentare la flessibilità e ridurre la **frammentazione esterna**.

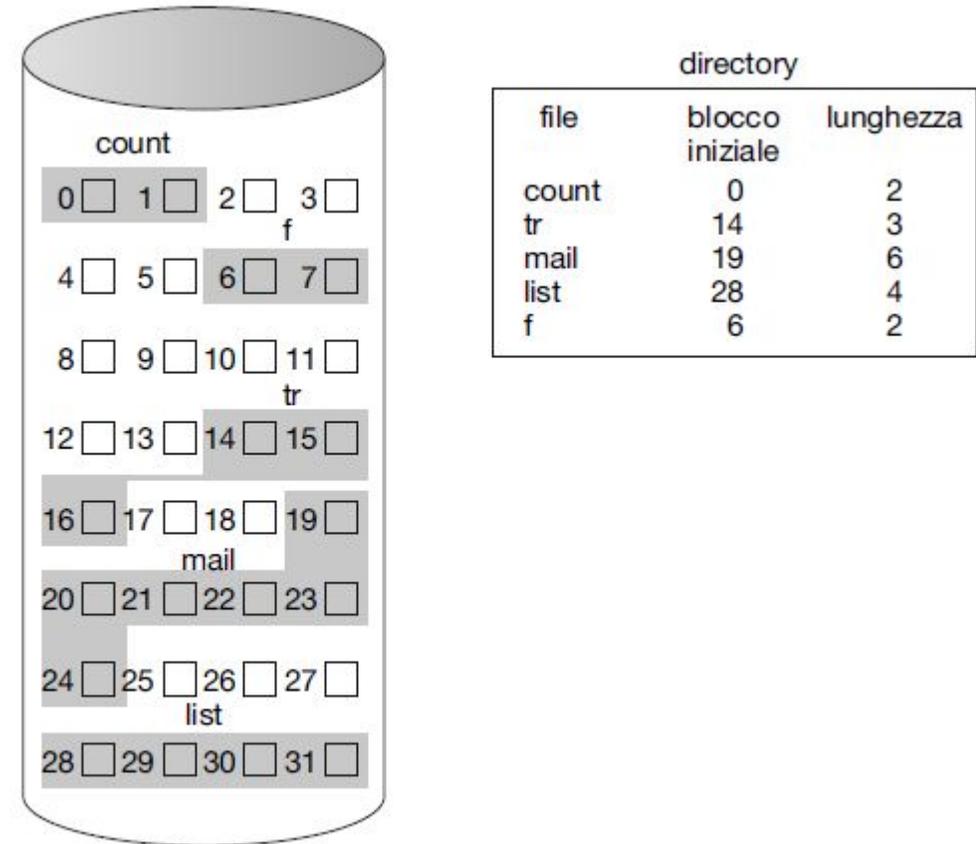


Figura 14.4 Allocazione contigua dello spazio dei dischi.

Allocazione concatenata

L'accesso diretto ai file è molto inefficiente con **l'allocazione concatenata**

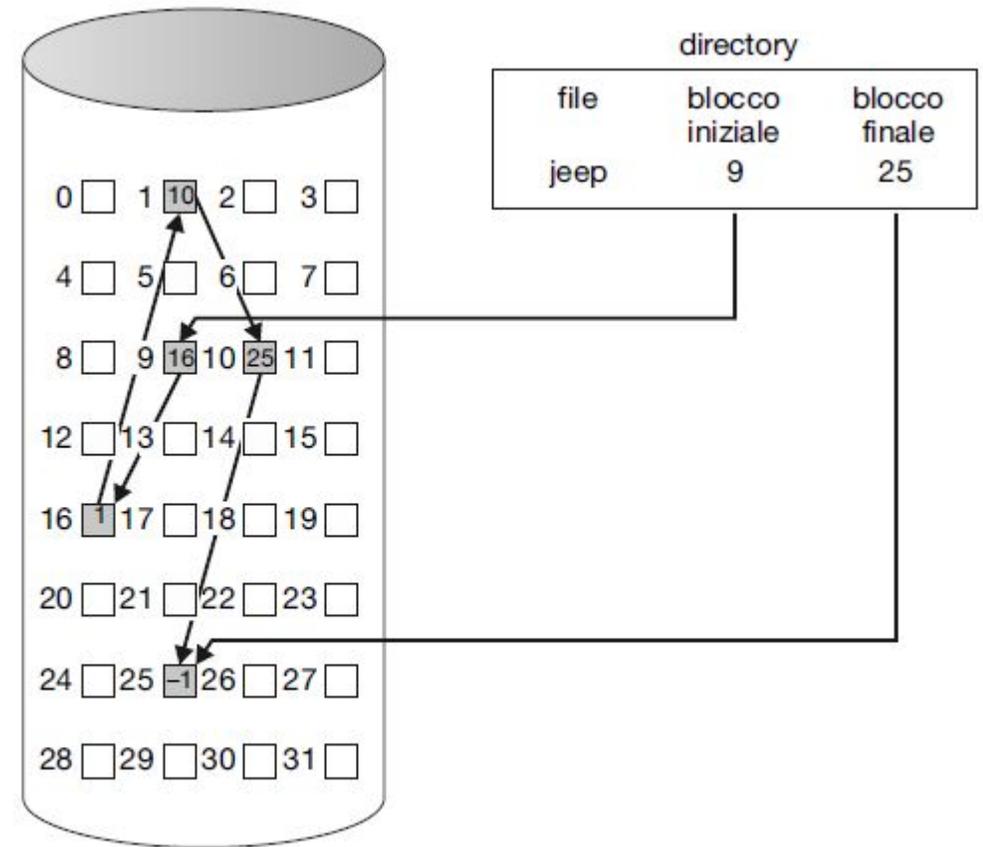


Figura 14.5 Allocazione concatenata dello spazio dei dischi.

Allocazione concatenata

Una variante importante del metodo di allocazione concatenata consiste nell'uso della **tabella di allocazione dei file** (*file allocation table, FAT*).

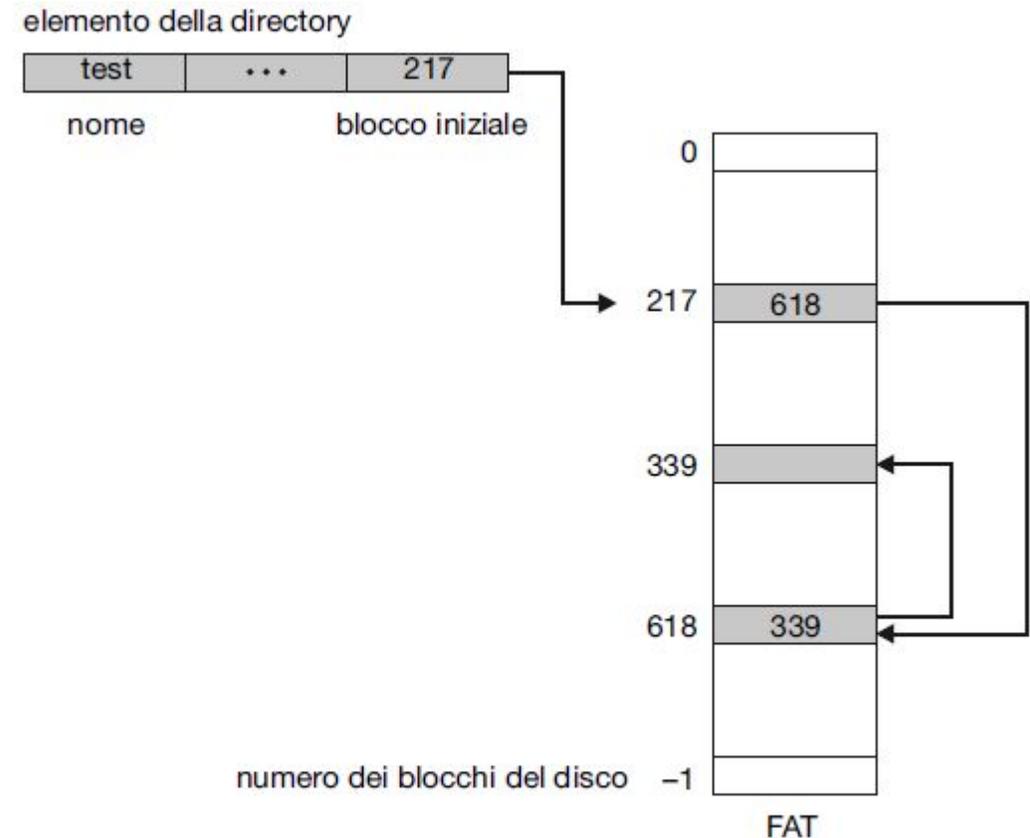


Figura 14.6 Tabella di allocazione dei file.

Allocazione indicizzata

L'**allocazione indicizzata** può richiedere un notevole overhead per il proprio **blocco indice**. L'allocazione indicizzata si può realizzare in **cluster** per incrementare il throughput e ridurre il numero di elementi dell'indice necessari. L'**indicizzazione in cluster** di grandi dimensioni è simile all'**allocazione contigua con estensioni**.

L'**allocazione indicizzata** raggruppa tutti i puntatori in una sola locazione: il **blocco indice**.

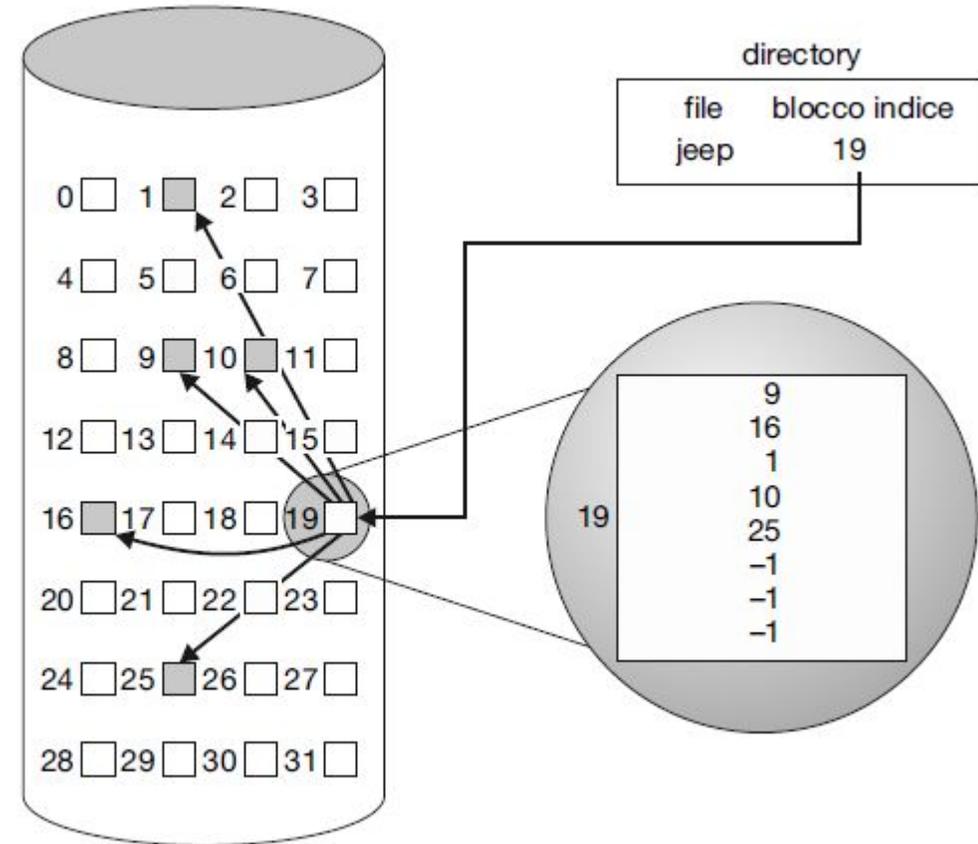


Figura 14.7 Allocazione indicizzata dello spazio dei dischi.

iNode

Fra i possibili meccanismi per gestire la questione della dimensione del **blocco indice** vi è il così detto **schema combinato** utilizzato nei sistemi basati su UNIX.

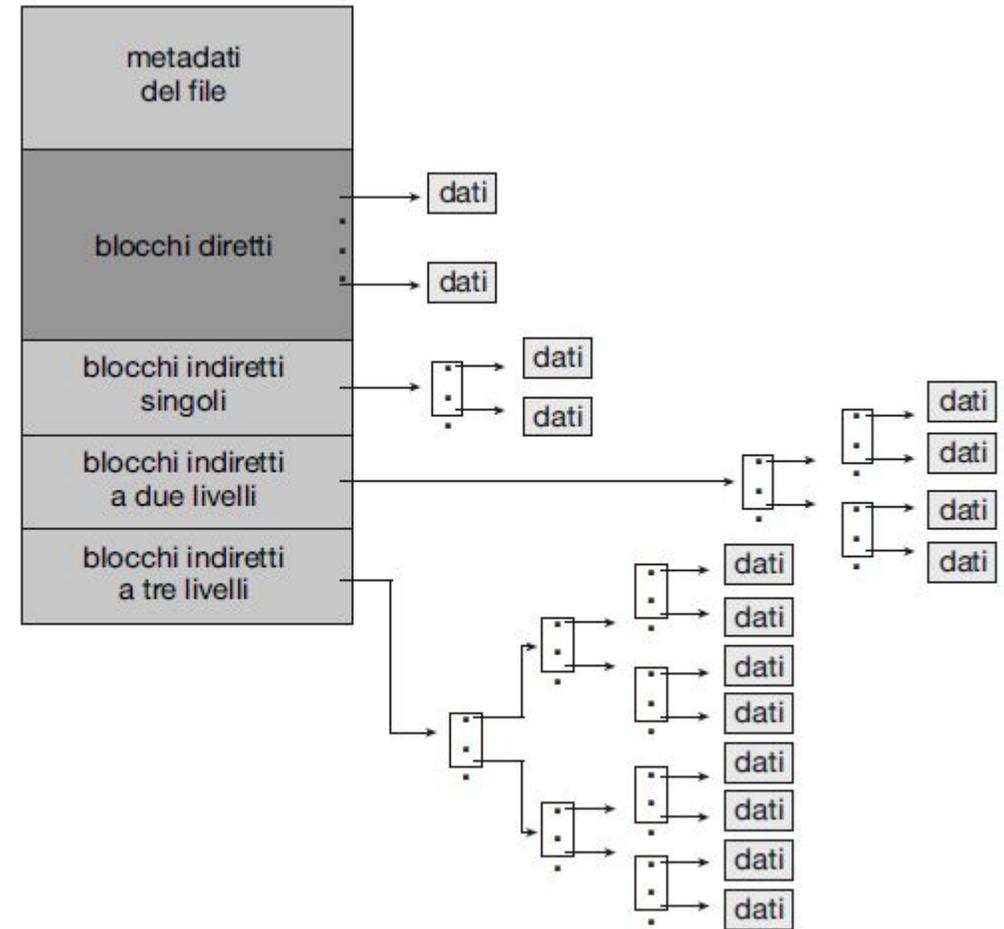


Figura 14.8 Inode di UNIX.

Gestione dello spazio libero

Per tener traccia dello **spazio libero** in un disco, il sistema conserva una **lista dello spazio libero**.

Metodi di allocazione dello spazio libero:

Vettori di bit

Liste concatenate

Ottimizzazioni:

Raggruppamento

Conteggio

FAT (colloca la lista concatenata in una singola area contigua)

Gestione dello spazio libero

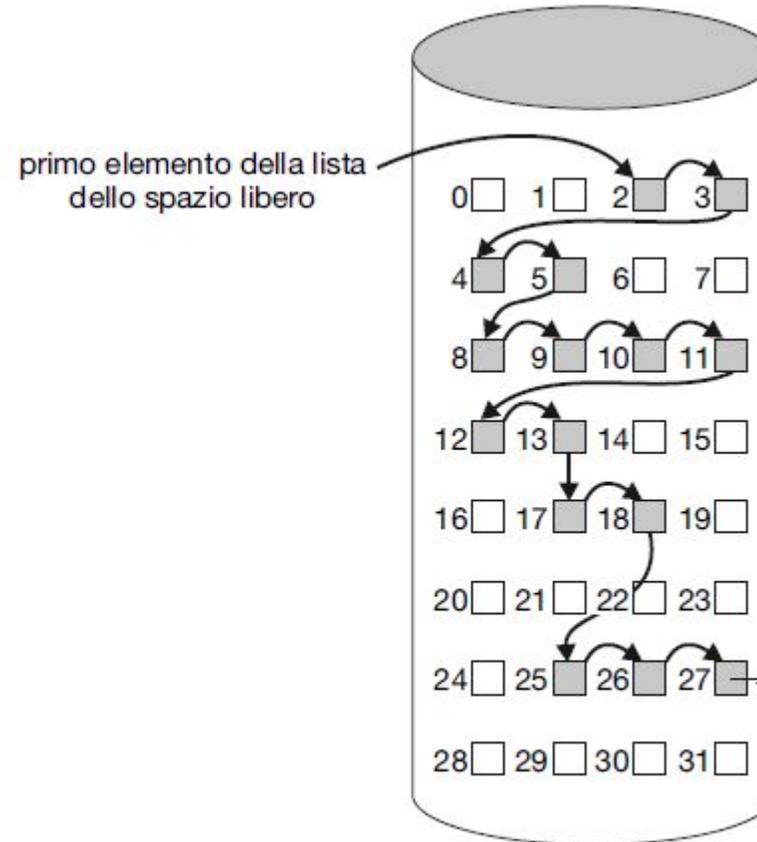


Figura 14.9 Lista concatenata degli spazi liberi su disco.

Efficienza e prestazioni

- Le procedure di gestione delle directory devono tener conto dell'**efficienza**, delle **prestazioni** e dell'**affidabilità**.
- **Buffer cache**: sezione separata della memoria centrale dove tenere i blocchi in previsione di un loro riutilizzo entro breve tempo.

Memory-mapping: prevede la lettura dei blocchi di disco dal file system e la loro memorizzazione nella **buffer cache**

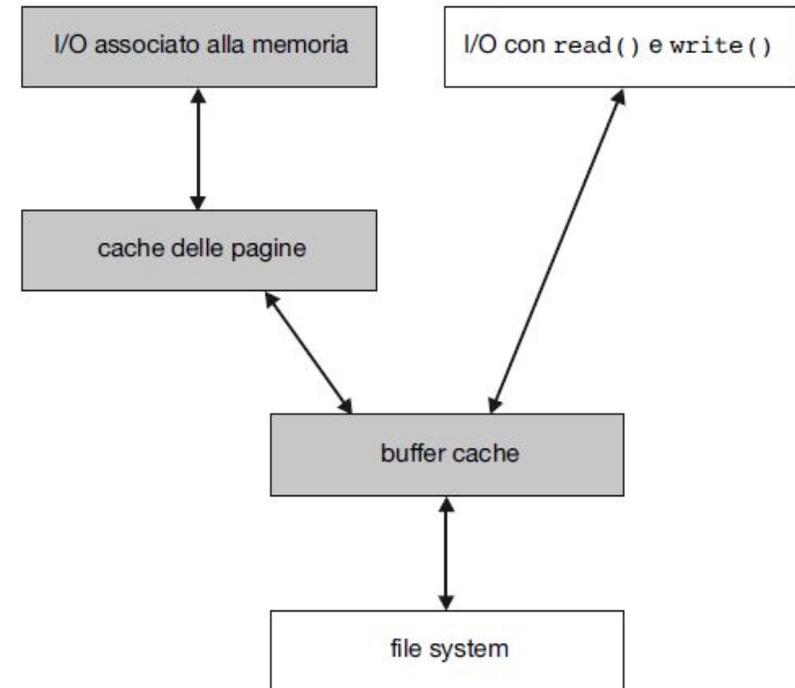


Figura 14.10 I/O senza una buffer cache unificata.

Buffer cache unificata

Con una **buffer cache unificata**, sia il memory-mapping sia le chiamate di sistema `read()` e `write()` usano la stessa cache delle pagine, con il vantaggio di evitare *// double caching* e di permettere al sistema di memoria virtuale di gestire dati del file system.

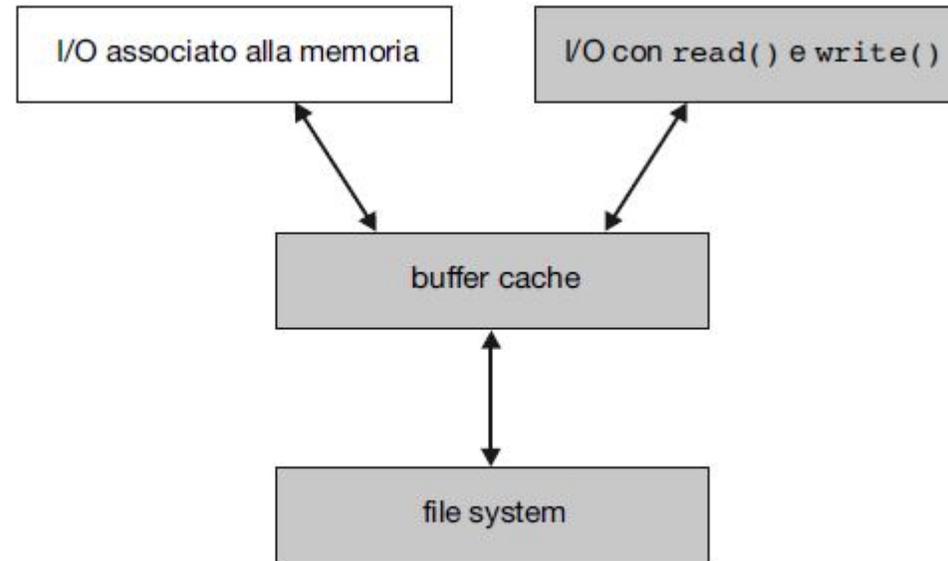


Figura 14.11 I/O con una buffer cache unificata.

Ripristino – verifica della coerenza

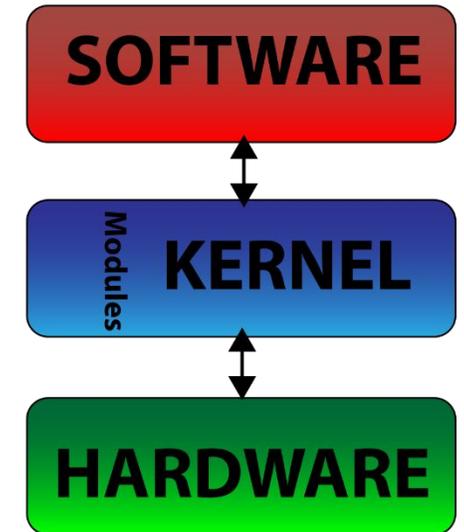
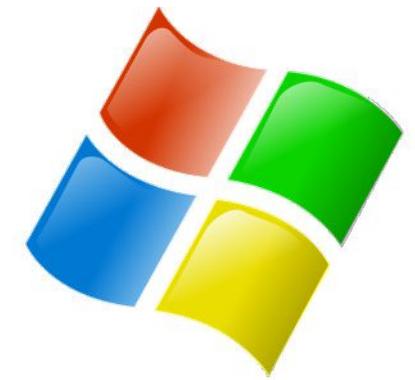
- Il **verificatore della coerenza** – un programma di sistema come fsck in UNIX – confronta i dati delle directory con i blocchi dati dei dischi, tentando di correggere ogni incoerenza.
- Può essere utilizzato per riparare la struttura danneggiata di un file system.
- Gli strumenti del sistema operativo per la creazione di **copie di backup** consentono la copiatura nelle unità a nastro dei dati contenuti nei dischi allo scopo di poterli **ripristinare** in seguito a perdite dovute a malfunzionamenti dei dispositivi fisici, errori del sistema operativo, o a errori degli utenti.



**UNIVERSITÀ DEGLI STUDI
DELLA BASILICATA**

Corso di Sistemi Operativi

File System



Docente:
Domenico Daniele
Bloisi

