Interfejs systemu plików

- Koncepcja pliku
- Metody i operacje dostępu do plików
- Struktura katalogowa
- Ochrona
- Semantyka spójności

Koncepcja pliku

- Pojęcie pliku:
  - logiczna jednostka magazynowania informacji
  - abstrakcyjny obraz informacji gromadzonej i udostępnianej przez system komputerowy.
  - nazwany zbiór powiązanych ze sobą informacji, zapisany w pamięci (pomocniczej)
  - POSIX: „ **File** - an object that can be written to, or read from, or both. A file has certain attributes, including access permissions and type. File types include regular file, character special file, block special file, FIFO special file, symbolic link, socket, and directory. Other types of files may be supported by the implementation”
- Zadania systemu operacyjnego:
  - zapewnienie odwzorowania pomiędzy abstrakcyjnym obrazem informacji a jego reprezentacją na urządzeniu fizycznym:
  - identyfikacja pliku (hierarchiczna struktura katalogów),
  - udostępnienie interfejsu operacji plikowych (API),
  - realizacja operacji dostępu do plików i katalogów z zapewnieniem bezpieczeństwa (synchronizacja i autoryzacja dostępu), spójności i efektywności.

Struktura pliku

- Brak struktury - ciąg słów, bajtów
- Zbiór prostych rekordów
  - wiersz (tekstu)
  - rekordy o stałej długości
  - rekordy o zmiennej długości
- Struktury złożone
  - sformatowany dokument
  - przemieszczalny plik wykonywalny
- Złożone struktury plików mogą być emulowane za pomocą struktury prostych przez wprowadzenie do zawartości pliku znaków sterujących.
- O strukturze pliku decyduje:
  - system operacyjny
  - program (programista aplikacji)

Atrybuty pliku

- **Nazwa** – jedyna informacja przechowywana w postaci czytelnej bezpośrednio przez człowieka.
- **Typ** – wymagany przez niektóre systemy operacyjne (dla interpretacji zawartości). Typ pliku może być rozpoznawany przez system, użytkownika bądź aplikację.
- **Położenie** – wskaźnik do urządzenia i położenia pliku na tym urządzeniu.
- **Rozmiar** – bieżący rozmiar pliku. **Uwaga**:

  1000B = 1kB; 1024B=1KB=1KiB (kibibyte);

  1000^2B=1MB; 1024^2B=1MiB (mebibyte)

- **Ochrona** – informacje służące do sprawdzania, kto może plik czytać, zapisywać, wykonywać itd.
- **Czas**, **data**, **identyfikator użytkownika** – dane służące do ochrony, bezpieczeństwa i doglądania użycia plików.

Informacje o plikach są przechowywane w strukturze katalogowej, znajdującej się na tym samym urządzeniu co pliki. Format pliku (→ typ) bywa zapisany w nazwie pliku, albo na początku pliku (***magic numbers***), np.:

ELF exec: 0x7F, 'ELF' , Unix script: '#!' ; MS-DOS exec: 'MZ' (0x4D, 0x5A).

Operacje plikowe

- **create** – tworzenie pliku
- **write** – zapisywanie do pliku
- **read** – czytanie z pliku
- **file seek** – zmiana bieżącej pozycji w pliku
- **delete** – usuwanie pliku (bądź jego dowiązania do pozycji katalogowej; znaczenie bywa różne)
- **truncate** – skracanie pliku
- **fd=open(F_i)** – znajduje w strukturze katalogowej dysku wpis pliku F_i i kopiuje zawartość tego wpisu do pamięci – jeśli pozwalają na to reguły dostępu dla danego procesu. Operacja tworzy nową sesję plikową. Deskryptor **fd** reprezentuje dostęp do pliku w ramach sesji plikowej.
- **close (*fd*)** – przepisuje zawartość struktury opisującej sesję plikową, związaną z deskryptodem **fd,** z pamięci do struktury katalogowej pliku (F_i) na dysku.

Blokady dostępu do plików

Systemy operacyjne często udostępniają procesom możliwość zakładania czasowej blokady dostępu do części bądź całego pliku:

- **Blokada obowiązkowa** (*mandatory file locking*) – jest wymuszana przez jądro; założenie takiej blokady powoduje, że system odmawia realizacji dostępu innym procesom przy próbie dostępu
- **Blokada doradzana** (*advisory file locking*) – nie jest wymuszana przez jądro; proces może sprawdzić, czy blokada jest założna przez inny proces, ale respektowanie blokady zależy od programisty

Otwarte pliki

System utrzymuje w pamięci operacyjnej szereg struktur danych służących do obsługi otwartych plików:

- **Wskaźnik bieżącej pozycji**, indywidualny dla każdej **sesji plikowej**
- **Licznik otwarć pliku** – pozwalający na usunięcie wpisu pliku z tablicy otwartych plików, gdy licznik osiąga wartość 0
- Kopia informacji pozwalającej na odszukanie zawartości pliku na urządzeniu fizycznym
- Struktura informująca o **prawach dostępu** do pliku

Sesja plikowa: ciąg operacji na pliku pomiędzy otwarciem, a zamknięciem dostępu do pliku. Sesja jest skojarzona z deskryptorem pliku, stanowi on identyfikator sesji plikowej.

Metody dostępu do pliku

- **Dostęp sekwencyjny** (*sequential access*)

  read next
  write next
  reset
  no *read after last write*
  (rewrite)

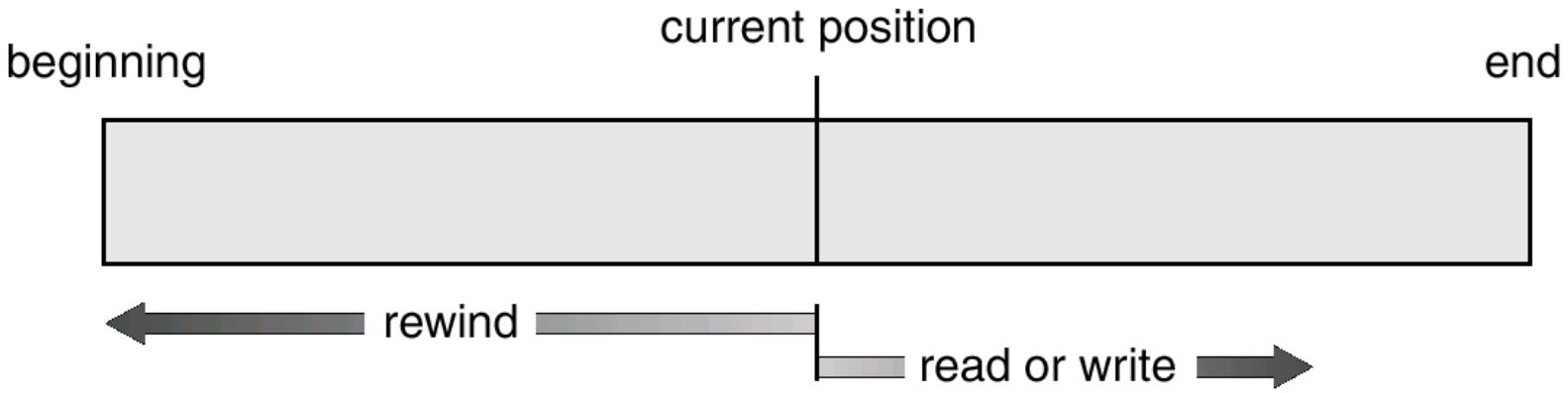
- **Dostęp bezpośredni** (*direct access*)

  read n
  write n
  position to n
  read next
  write next
  rewrite n

  n = numer bloku w pliku (względny)

Metody dostępu do pliku – c.d.

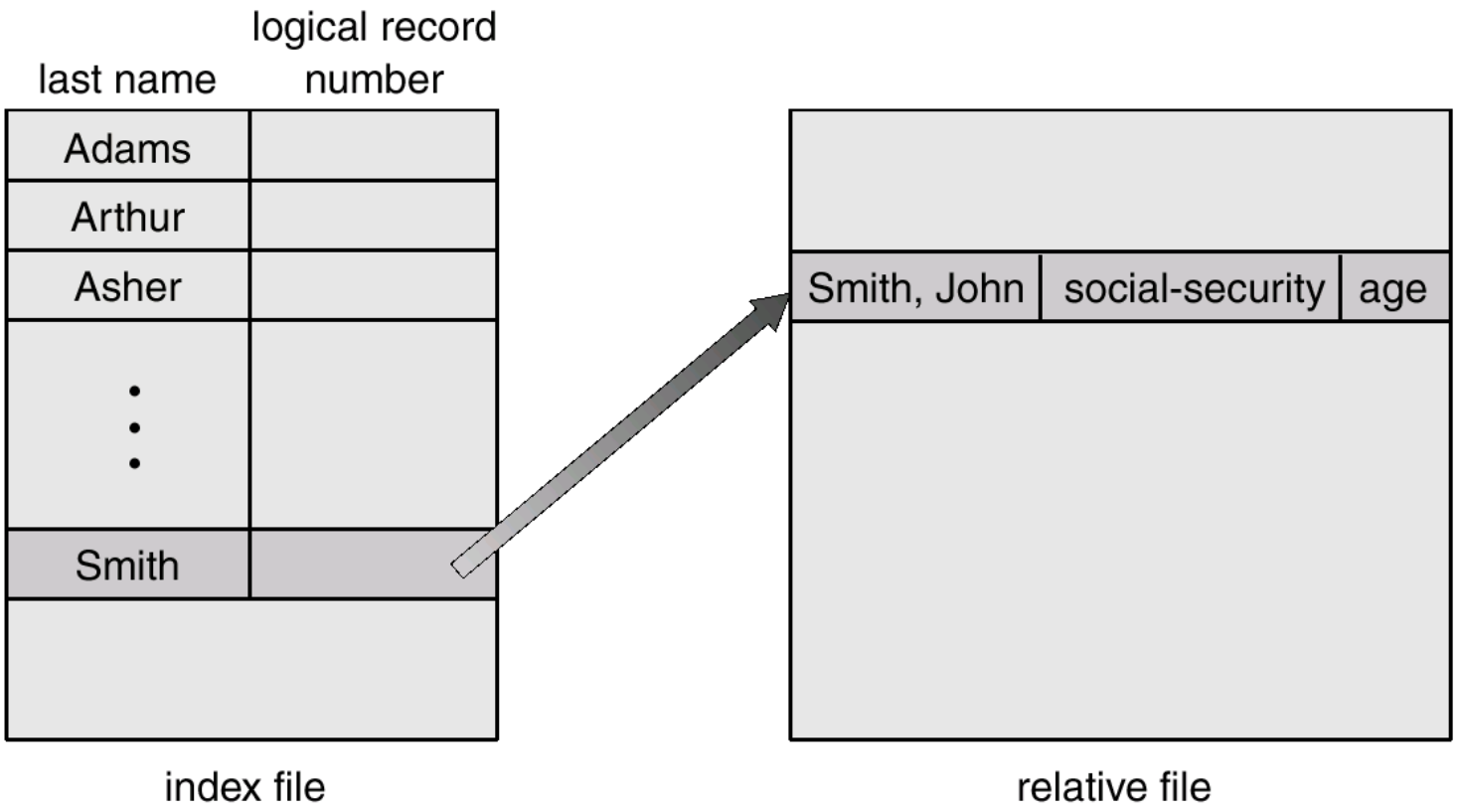
- Dostęp sekwencyjny



- Symulacja dostępu sekwencyjnego za pomocą dostępu bezpośredniego

sequential access	implementation for direct access
reset	*cp* = 0;
read next	*read cp*; *cp* = *cp*+1;
write next	*write cp*; *cp* = *cp*+1;

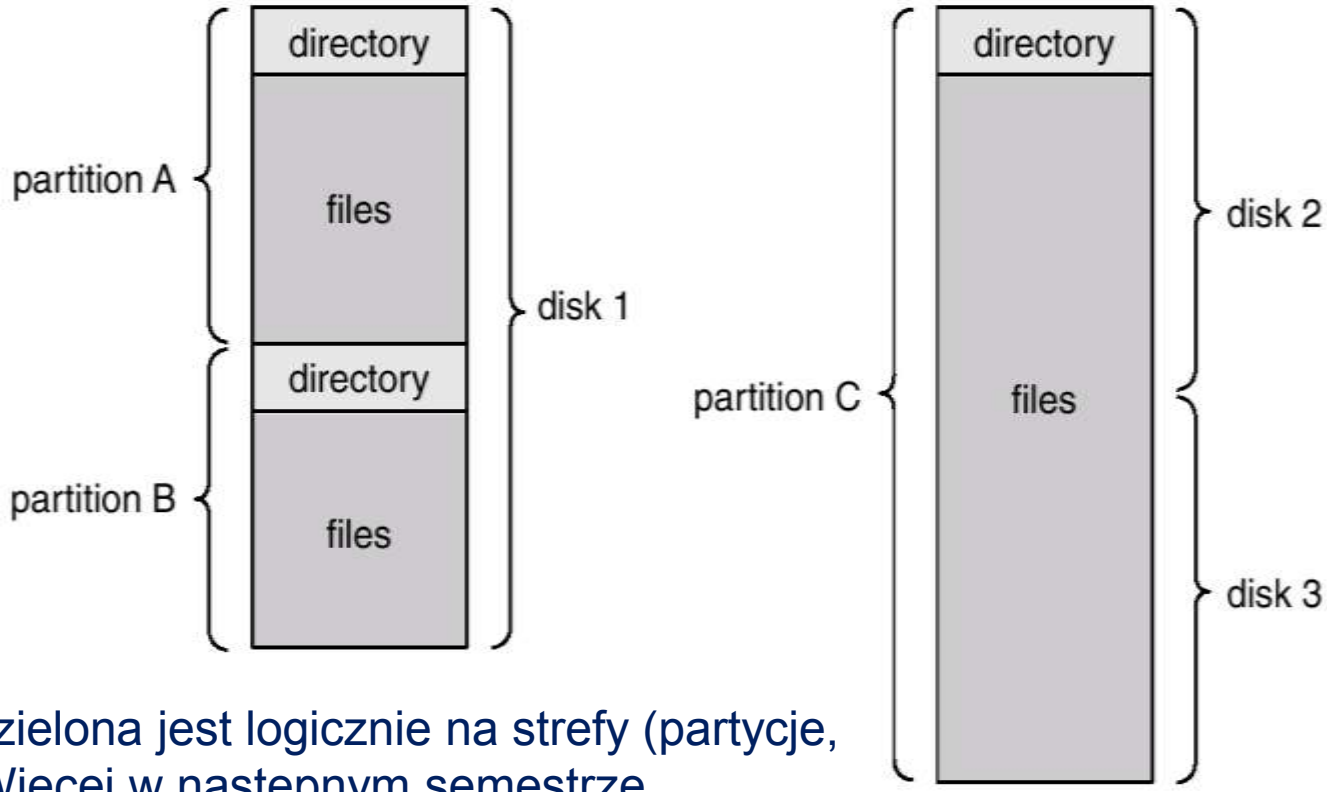
Przykład wykorzystania indeksu



Plik indeksowy

Plik spowinowacony

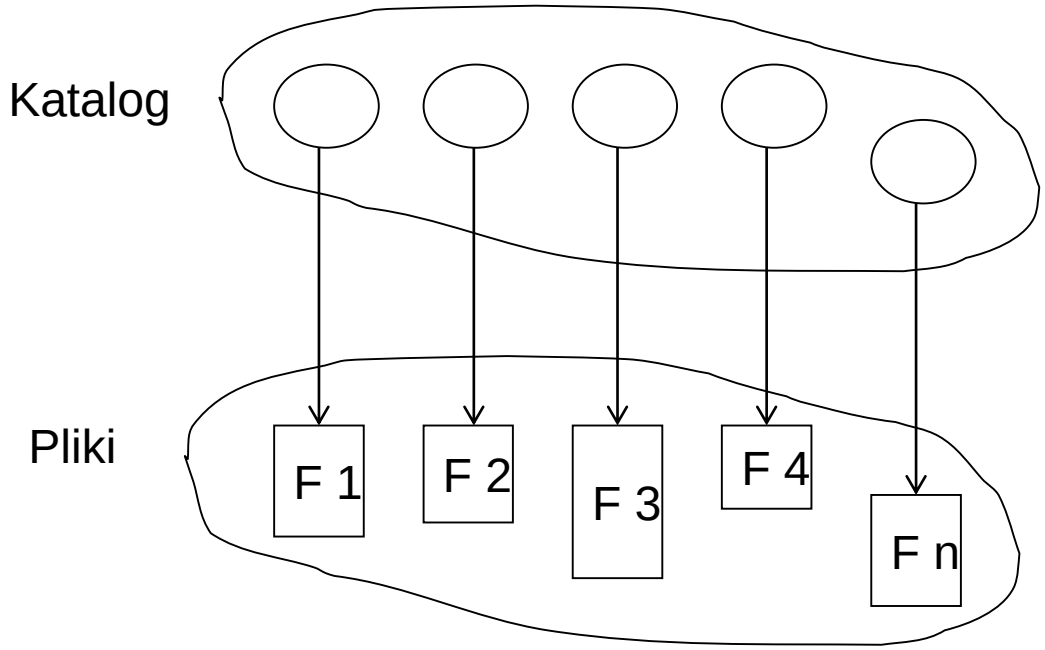
Typowa organizacja systemu plików



- Pamięć urządzeń podzielona jest logicznie na strefy (partycje, wolumeny, tomy,…). Więcej w następnym semestrze.
- Dyski/partycje mogą być zabezpieczane przed skutkami awarii (RAID).
- Dyski/partycje mogą być używane „na surowo” bądź po sformatowaniu.
- Sformatowana strefa zawiera nagłówek, główny katalog oraz miejsce na podkatalogi i pliki.

Struktura katalogowa plików

- Zarówno struktura katalogowa jak i pliki są przechowywane na tym samym urządzeniu.
- Kopie struktury katalogowej i plików są przechowywane w archiwach kopii zapasowych, np. na taśmach magnetycznych, płytach CD-ROM itd., umożliwiając odtwarzanie wcześniejszego stanu systemu plików.



Organizacja logiczna katalogów

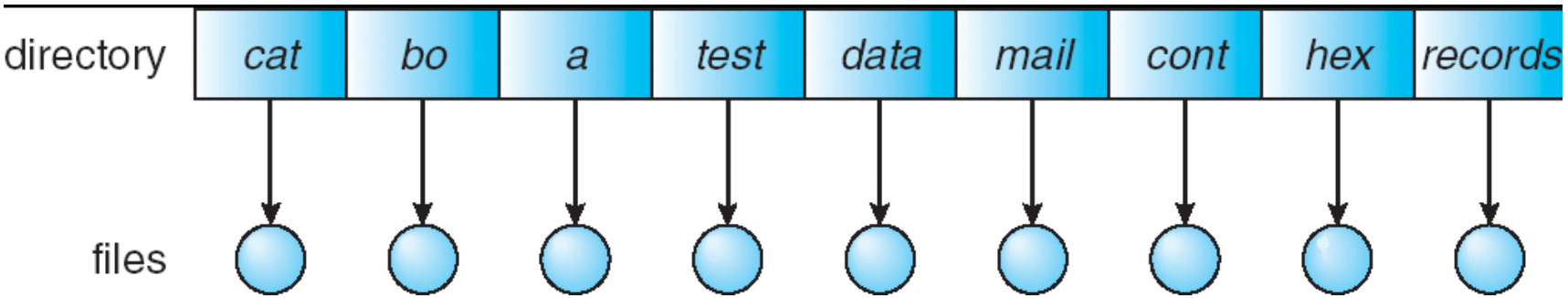
Organizacja katalogów powinna mieć na względzie:

- efektywność operacji katalogowych. Przykładowe operacje:
  - odnajdywanie pliku (rozkład ścieżki dostępu) – najczęstsza
  - tworzenie pliku
  - usuwanie pliku
  - przemianowywanie pliku
  - obchód systemu plików

- wymagania nazewnicze – odpowiednie do wymagań użytkowników.
  - Dwóch użytkowników może używać tej samej nazwy dla różnych plików..
  - Ten sam plik może mieć kilka różnych nazw (synonimów).
- wymagania grupowania – powinno być możliwe grupowanie plików wg ich funkcji, własności (np., programy w języku Pascal, gry, …)

Katalog jednopoziomowy

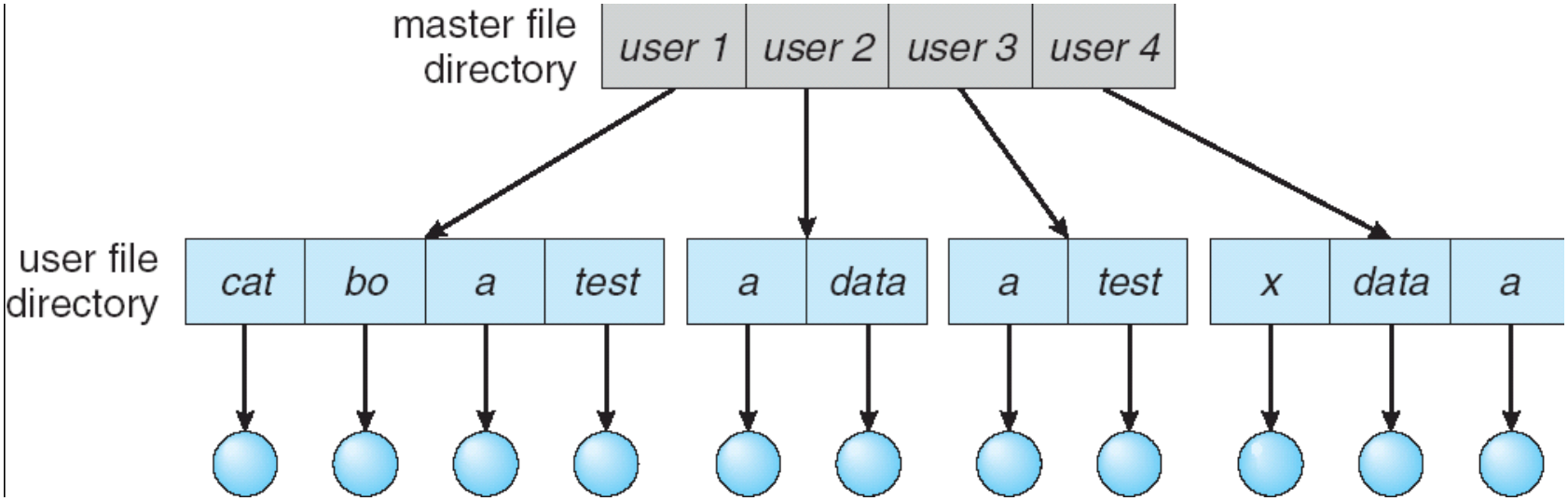
- Jeden katalog dla wszystkich użytkowników.



- Problem nazewniczy
- Problem grupowania plików

Katalog dwupoziomowy

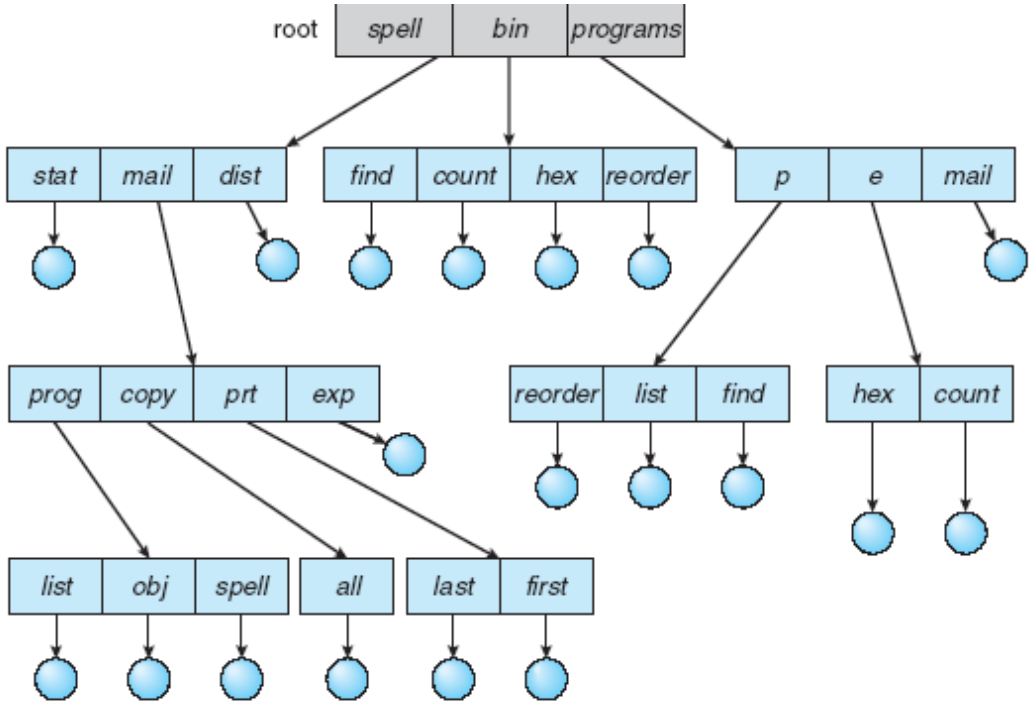
- Oddzielny katalog dla każdego użytkownika.



- Nazwa ścieżki (*path name)*
- Różni użytkownicy mogą użyć tej samej nazwy (dla różnych plików)
- Efektywne przeszukiwanie katalogu użytkownika
- Problem grupowania plików nierozwiązany

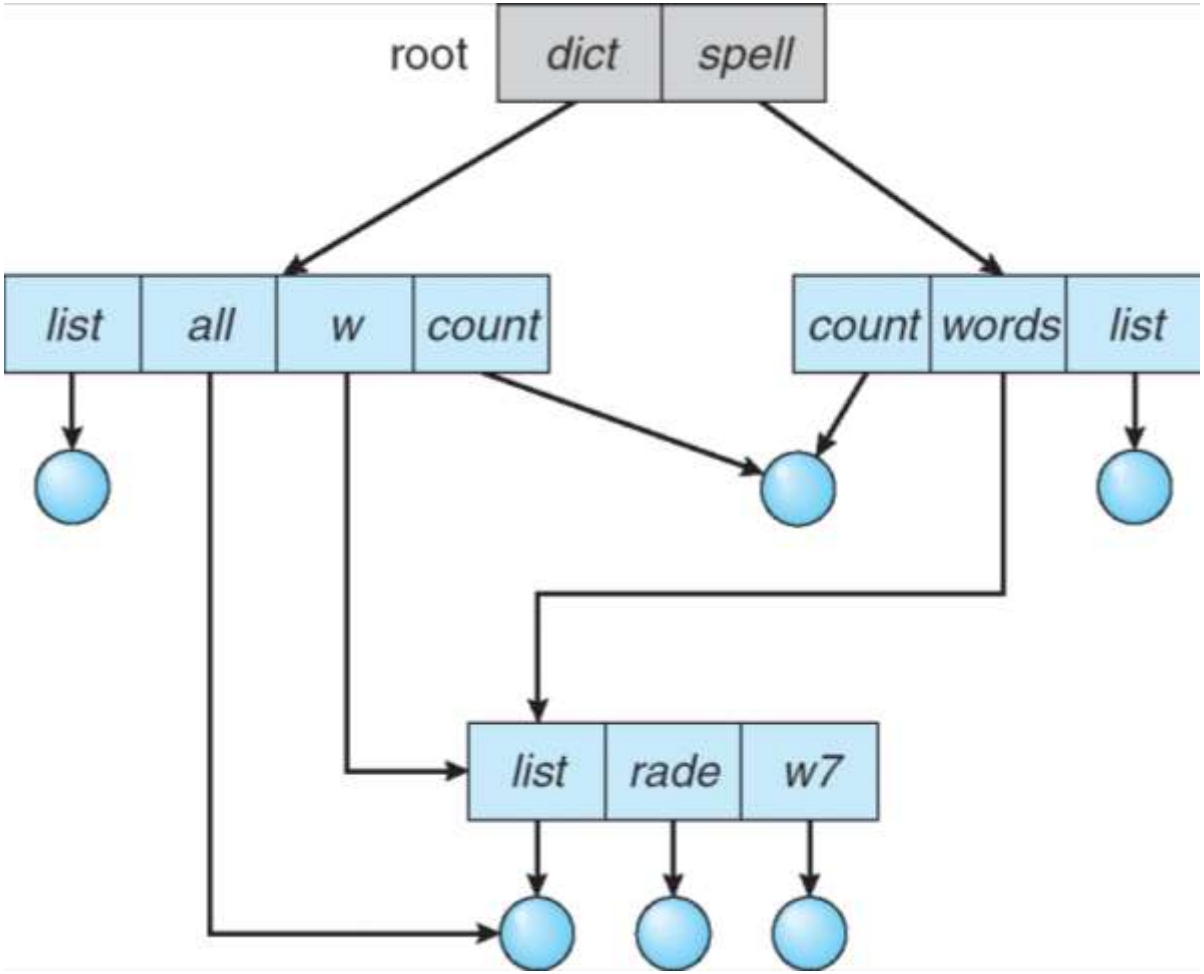
Katalog o strukturze drzewa

- Efektywne przeszukiwanie
- Rozwiązany problem grupowania plików
- Katalog bieżący (current directory, working directory)
  - **cd** /spell/mail/prog
  - **type** list
- **Ścieżki** (dostępu do pliku **względne** bądź **bezwzględne**
- Usunięcie katalogu powoduje usunięcie całej poddrzewa poniżej niego Np. usunięcie katalogu ”mail” powoduje utratę dwóch 4-ch podkatalogów 6-i plików →



Katalog o strukturze acyklicznego grafu

- Możliwe jest dzielenie dostępu do plików i katalogów.

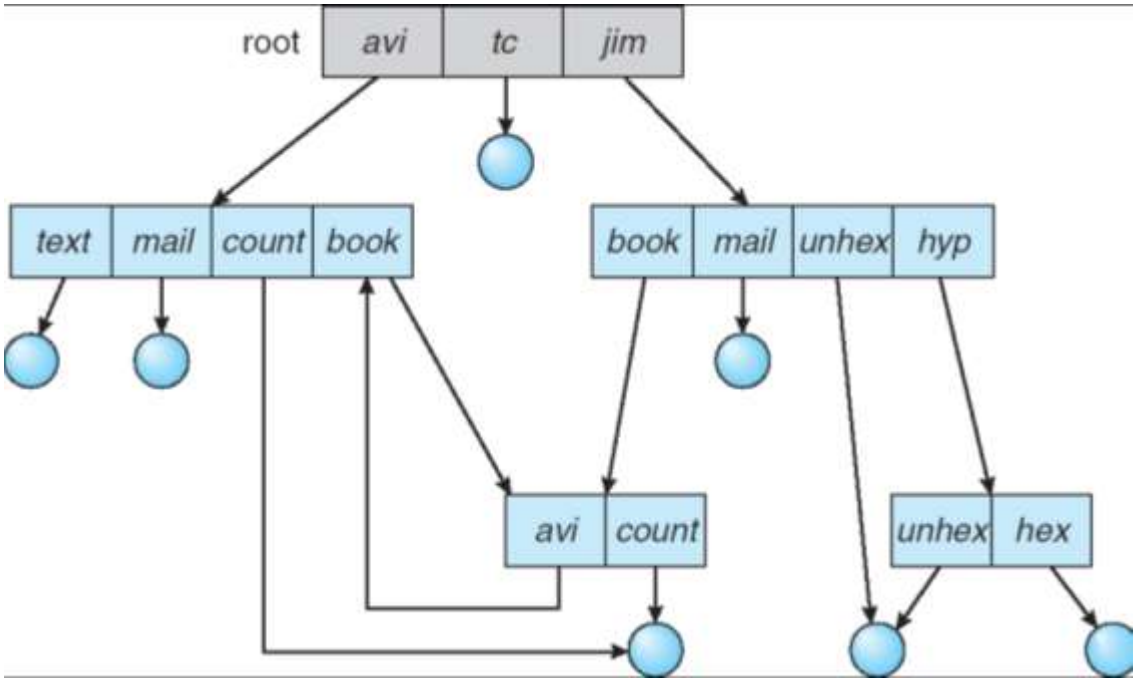


Katalog o strukturze acyklicznego grafu (c.d.)

- Możliwe jest utworzenie dwóch nazw (ścieżek) do tego samego pliku czy katalogu (*aliasing*)
- Jeśli usunąć jedną ścieżkę dostępu do pliku (katalogu) wraz z tym plikiem (katalogiem) $\Rightarrow$ problem: inne ścieżki dostępu do tego pliku (katalogu) przestają być ważne (*dangling pointers)*.

  Rozwiązania:
  - Wykaz odniesień w postaci tablicy wskaźników zwrotnych: od pliku do wszystkich katalogów nań wskazujących. Niewygoda - zmienna długość rekordu opisującego plik.
  - Modyfikacja: lista wskaźników zamiast tablicy wskaźników zwrotnych.
  - Licznik odniesień zamiast wykazu odniesień.

Graf ogólny katalogów

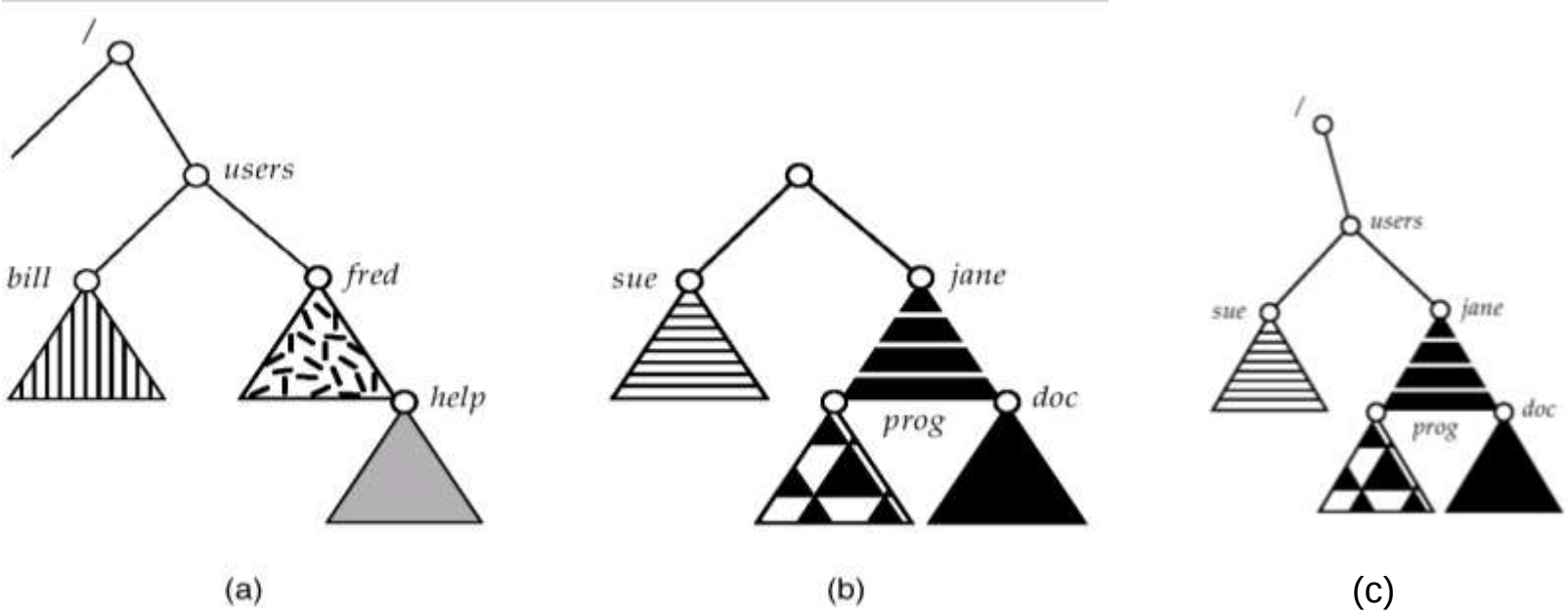


Jak zagwarantować, że nie ma cykli?

- Zezwalać na dowiązania (*link)* do plików a nie do katalogów
- Odśmiecać system plików (*garbage collection*).
- Przy każdym tworzeniu dowiązania uruchamiać algorytm wykrywania cykli.

Montowanie podsystemu plików

- System plików musi być **montowany** w systemie operacyjnym zanim może być użyty (rys. a)
- Niezamontowany podsystem plików (rys. b) jest montowany w **punkcie montażu** (*mount point*) – na rys. (c) ma on ścieżkę dostępu: ***/*users**). Poprzednia zawartość katalogu users jest przysłonięta przez zamontowane poddrzewo plików. Zazwyczaj podsystem plików montuje się do pustego katalogu.



Semantyka spójności

- **Dzielenie dostępu** do plików jest pożądane w systemach wieloużytkownikowych. W systemach rozproszonych pliki mogą być dostępne z różnych punktów sieci – np. za pomocą protokołu Network File System (NFS), czy Common Internet File System (CIFS).
- **Semantyka spójności** określa sposób realizacji jednoczesnego dostępu do pliku dzielonego przez wielu użytkowników.
- Przykłady realizacji semantyki spójności:
  - **Semantyka systemu UNIX**: plik ma jeden obraz, a wynik zapisu jest natychmiast widoczny dla innych użytkowników; istnieje tryb dzielenia, w którym użytkownicy wspólnie korzystają ze wskaźnika bieżącego położenia w pliku.
  - **Semantyka sesji**: wynik zapisu nie jest natychmiast widoczny dla innych użytkowników, a po zakończeniu sesji
  - **Semantyka stałych plików dzielonych** (zawartość niezmienna).
- **Rozproszone systemy plików** wymagają obsługi nowych rodzajów awarii – spowodowanych przez awarię sieci czy serwera. Wydobywanie z awarii jest łatwiejsze w systemach w których klient, a nie serwer przechowuje informacje o stanie dostępu.

Ochrona

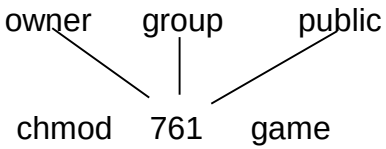
- Właściciel/twórca pliku (bądź osoba odpowiedzialna za bezpieczeństwo) powinien móc kontrolować: co może być zrobione z plikiem i przez kogo
- Rodzaje dostępu
  - Czytanie (*Read*)/pisanie (*Write*)/eykonywanie (*Execute*)
  - Dopisywanie (*Append*), usuwanie (*Delete*), opisywanie (*List*), tj. wyprowadzenie nazwy i atrybutów pliku
- Najczęściej uzależnia się dostęp do plików od identyfikacji użytkownika, bądź jego roli (*role-based access controll)*
- *Linux cgroups* – jądro ogranicza dostęp do wybranych zasobów dla grup procesów. Procesy różnych grup mogą widzieć różne składniki fizycznego systemu.
- Najogólniejsza metoda implementacji dostępu zależnego od tożsamości polega na skojarzeniu z każdym plikiem i katalogiem wykazu dostępów (*access list*), zawierającego identyfikatory użytkowników i dozwolone rodzaje dostępu. Dla zmniejszenia wykazu wprowadza się klasy (grupy) użytkowników.

Grupy dostępów (UNIX)

- Tryby dostępu: read, write, execute
- Trzy klasy użytkowników:

			RWX
a) dostęp właściciela	7	⇒	1 1 1
b) dostęp grupy	6	⇒	1 1 0
c) dostęp publiczny	1	⇒	0 0 1

- Administrator może utworzyć nazwaną grupę użytkowników, np. G, i przedzielić do niej wybranych użytkowników.
- Dla danego pliku (np. *game*) lub katalogu można zezwolić na dostęp dla tej grupy, np...



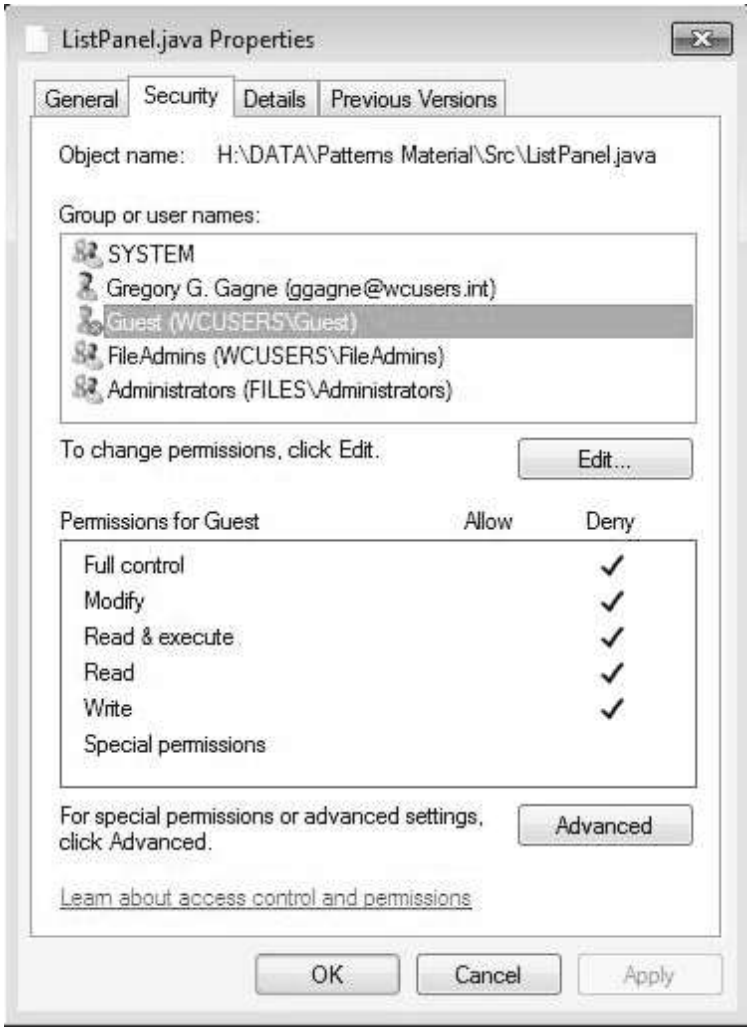
- Dostęp dla grupy G uzyskuje się za pomocą:

chgrp *G* *game*

Przykładowy listing zawartości katalogu w systemie UNIX

-rw-rw-r--	1 pbg	staff	31200	Sep 3 08:30	intro.ps
drwx------	5 pbg	staff	512	Jul 8 09.33	private/
drwxrwxr-x	2 pbg	staff	512	Jul 8 09:35	doc/
drwxrwx---	2 pbg	student	512	Aug 3 14:13	student-proj/
-rw-r--r--	1 pbg	staff	9423	Feb 24 2003	program.c
-rwxr-xr-x	1 pbg	staff	20471	Feb 24 2003	program
drwx--x--x	4 pbg	faculty	512	Jul 31 10:31	lib/
drwx------	3 pbg	staff	1024	Aug 29 06:52	mail/
drwxrwxrwx	3 pbg	staff	512	Jul 8 09:35	test/

Zarządzanie dostępem do plików w Windows 7



Do omówienia w przyszłym semestrze