File System

د. فادي تركاوي

File Interface
واجهة التخاطب

- open, close

- فتح، إغلاق

- open returns a file identifier (or handle or descriptor), which is used in subsequent operations to identify the file.

- فتح الملف يعيد محدد الملف الذي يستخدم للعمليات على الملف

- read, write
  - must specify which file to read, which part of the file to read, and where to put the data that has been read (similar for write).

- قراءة، كتابة

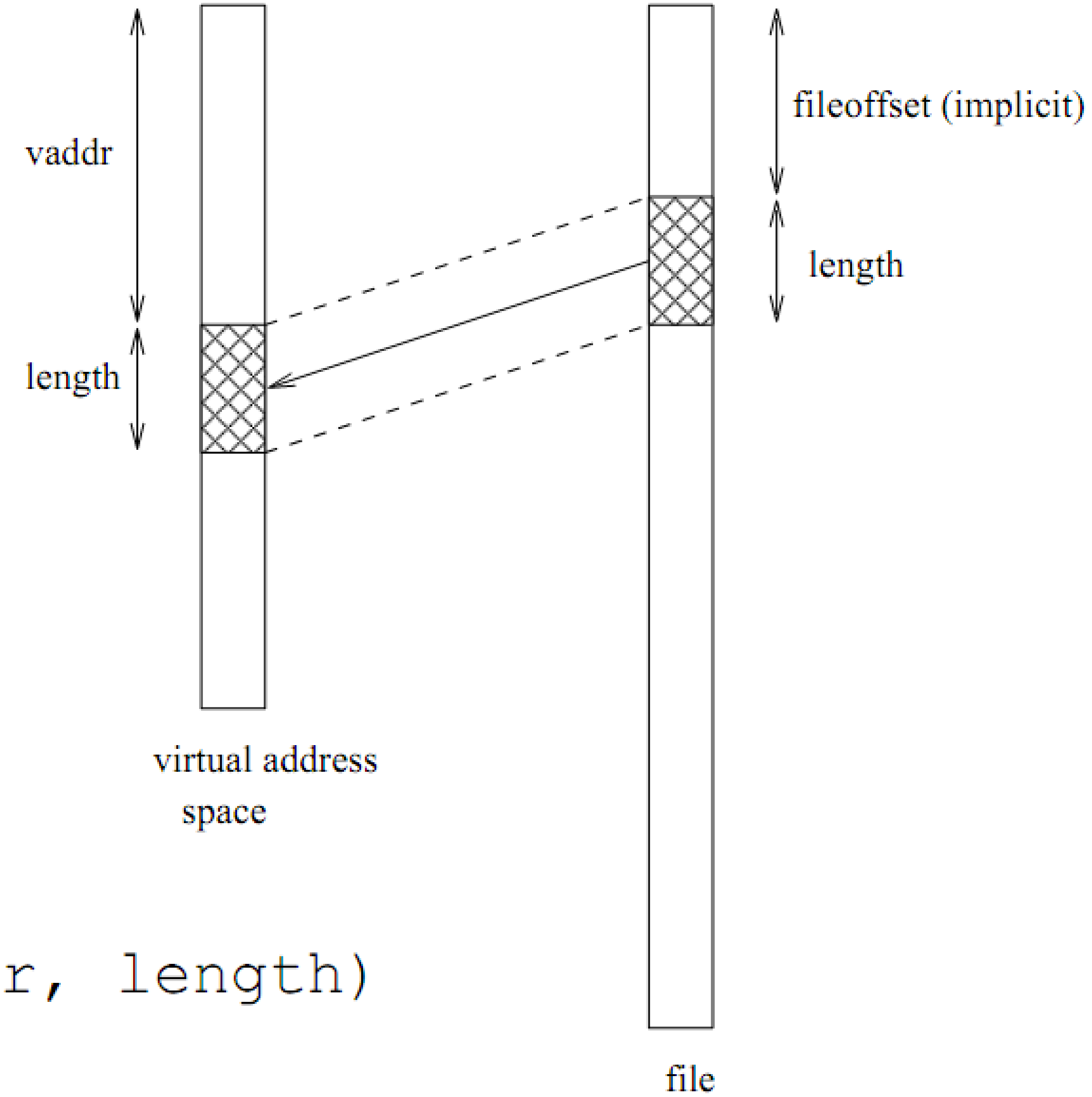
- Seek

- البحث

- get/set file attributes, e.g., Unix fstat, chmod

- توضيع خصائص الملف

File Read

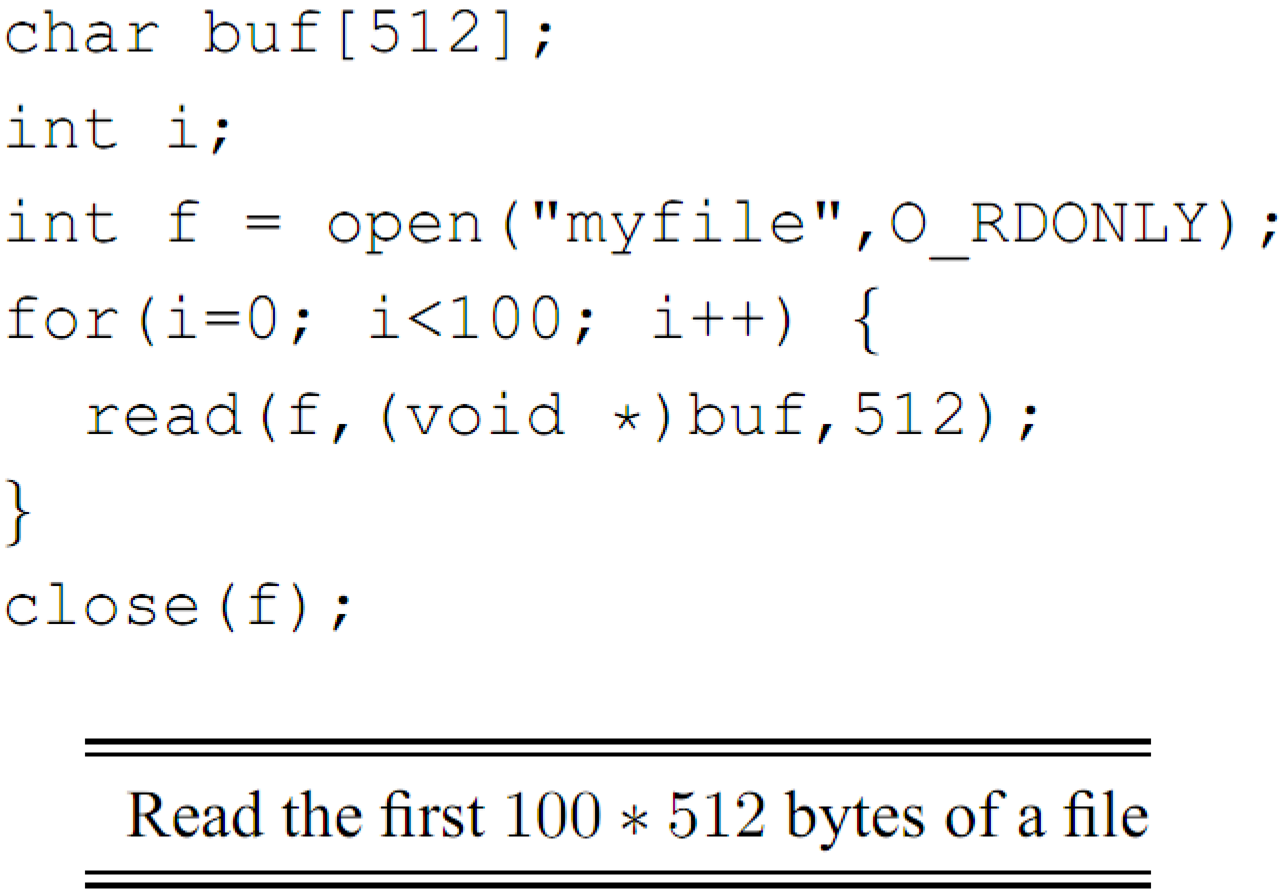


```
read(fileID, vaddr, length)
```

File Position

- may be associated with the file, with a process, or with a file descriptor (Unix style)
- read and write operations
  - start from the current file position
  - update the current file position
- this makes sequential file I/O easy for an application to request
- for non-sequential (random) file I/O, use:
  - seek, to adjust file position before reading or writing
  - a positioned read or write operation, e.g., Unix pread, pwrite: (القراءة بتحديد موقع)
  - pread(fileId,vaddr,length,filePosition)

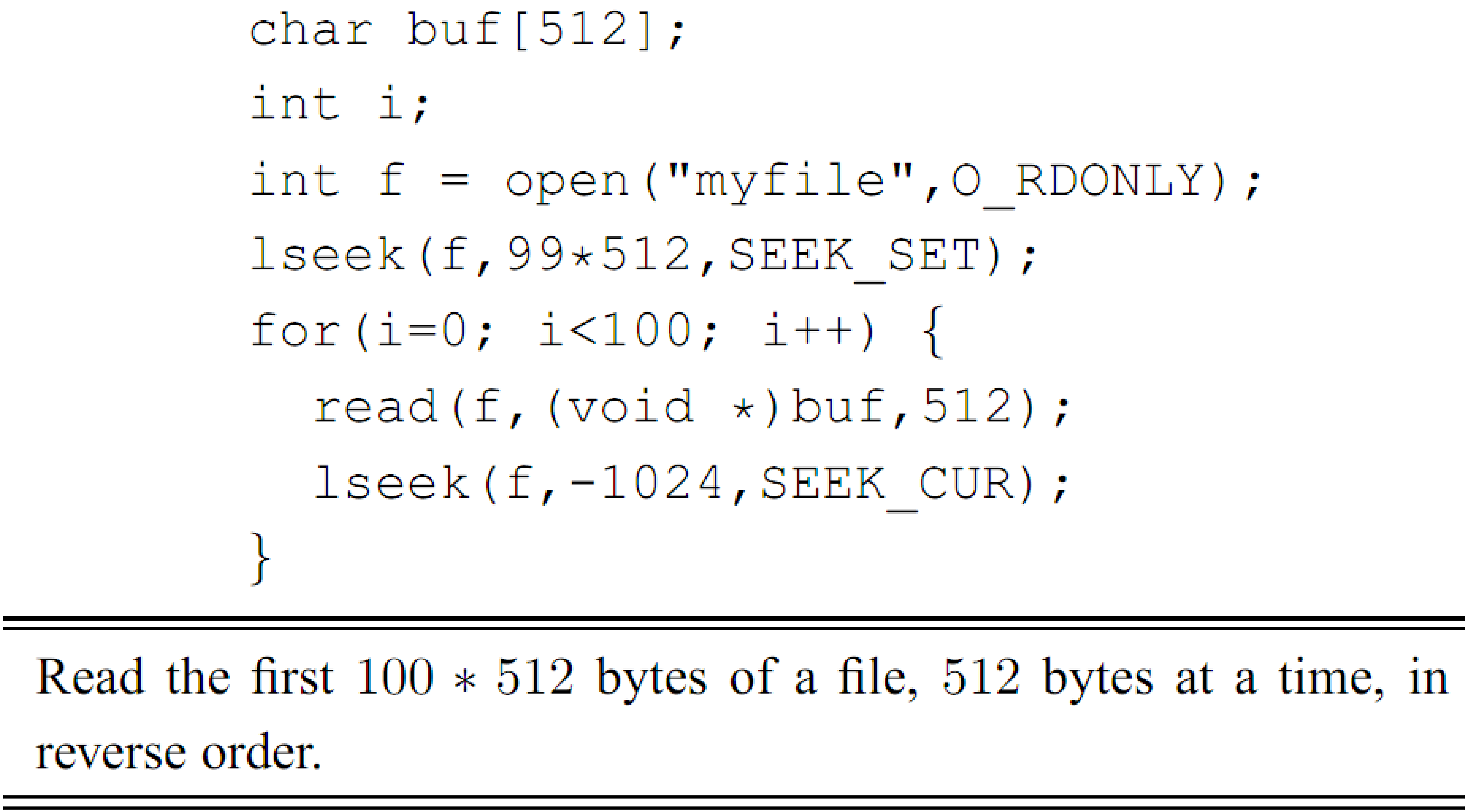
Sequential File Reading Example (Unix)



```
char buf[512];
int i;
int f = open("myfile",O_RDONLY);
for(i=0; i<100; i++) {
   read(f,(void *)buf,512);
}
close(f);
```

Read the first 100 * 512 bytes of a file

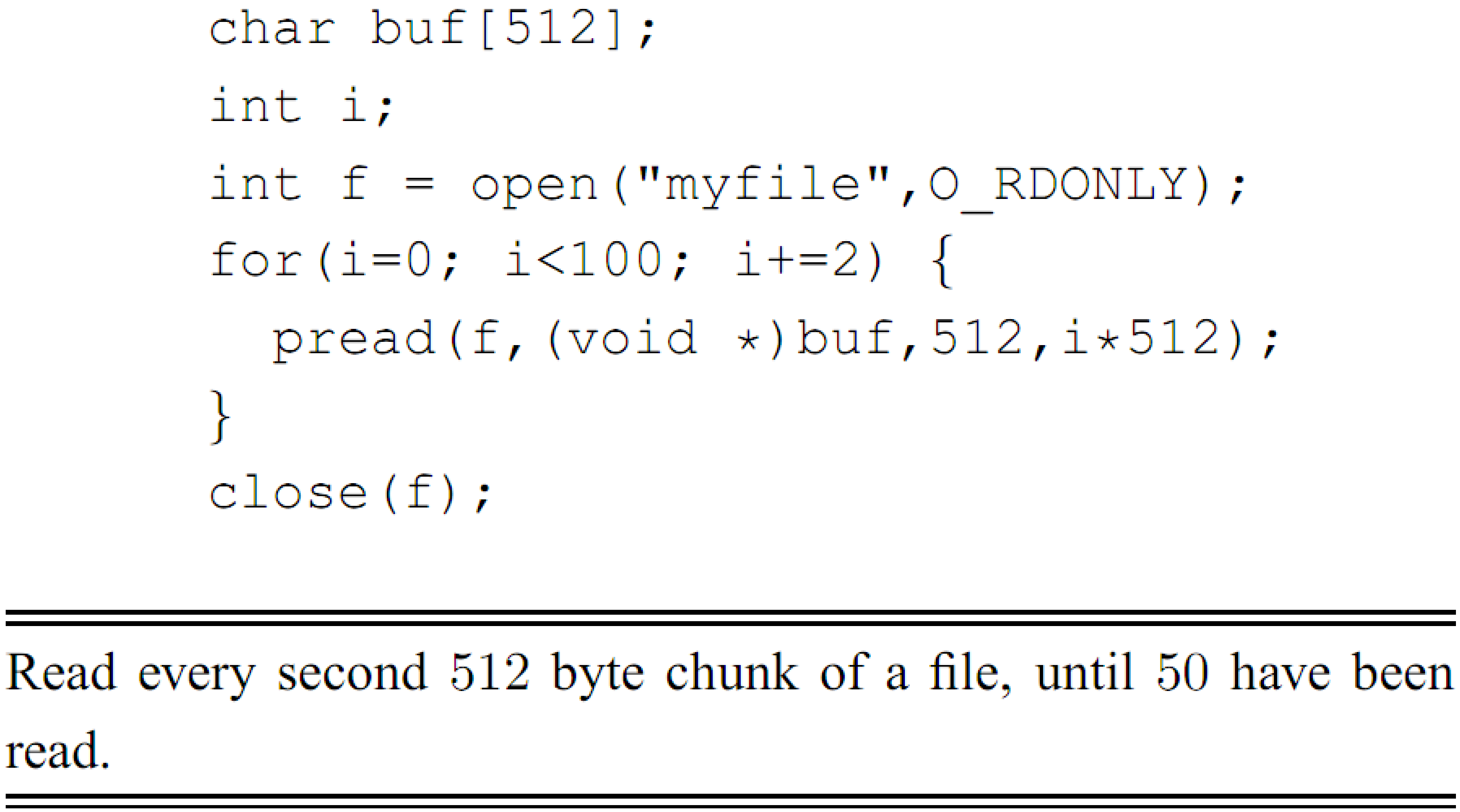
File Reading Example Using Seek (Unix)



```
char buf[512];
int i;
int f = open("myfile",O_RDONLY);
lseek(f,99*512,SEEK_SET);
for(i=0; i<100; i++) {
  read(f,(void *)buf,512);
  lseek(f,-1024,SEEK_CUR);
}
```

Read the first $100 * 512$ bytes of a file, 512 bytes at a time, in reverse order.

File Reading Example Using Positioned Read



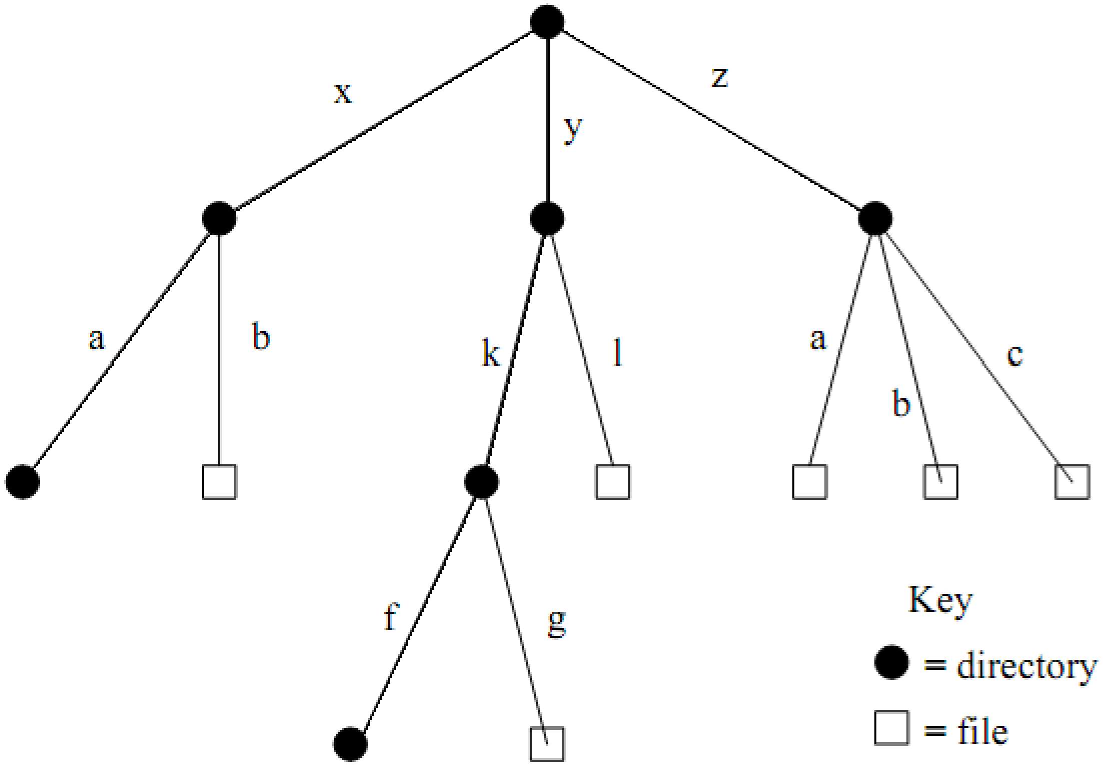
```
char buf[512];
int i;
int f = open("myfile",O_RDONLY);
for(i=0; i<100; i+=2) {
  pread(f,(void *)buf,512,i*512);
}
close(f);
```

Read every second 512 byte chunk of a file, until 50 have been read.

File Names

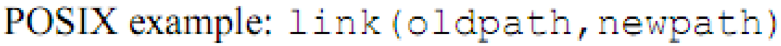
- أغراض التطبيقات المرئية مثل الملفات و المجلدات تعطى أسماء
- إن نظام الملفات مسؤول عن تخصيص أسماء للأغراض
- مجال الأسماء عادة يكون بنيوي هرمي غالبا شجرة أو غراف موجه غير دائري DAG
- مجال الأسماء namespace يؤمن طرق من أجل مستخدمين و تطبيقات تنظم و تدير المعلومات
- في مجال الاسماء البينوي، الأغراض ربما تكون محدد من خلال pathnames التي توصف الممر من الجذر إلى الغرض الذي نحدده مثال:
- /home/kmsalem/courses/cs350/notes/filesys.ps

Hierarchical Namespace Example



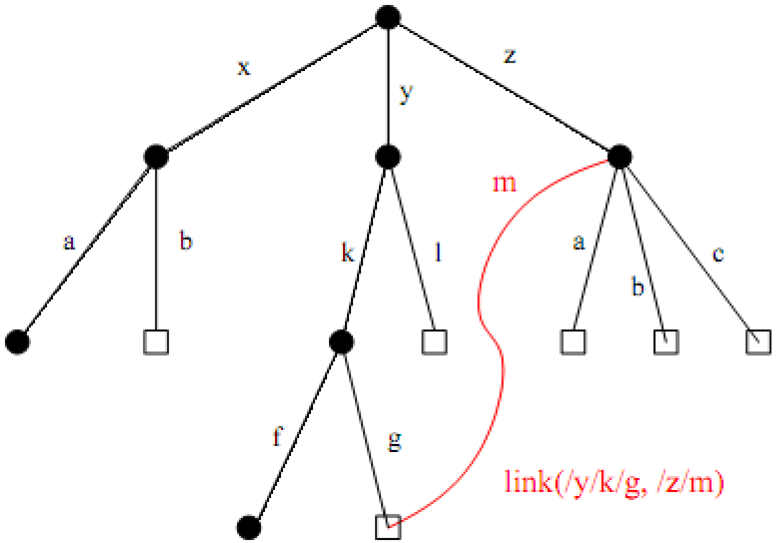
Hard Links

- a hard link is an association between a name and an underlying file (or directory)
- typically, when a file is created, a single link is created to the that file as well (else the file would be difficult to use!)
  - POSIX example: creat(pathname,mode) creates both a new empty file object and a link to that object (using pathname)



POSIX example: `link(oldpath,newpath)`

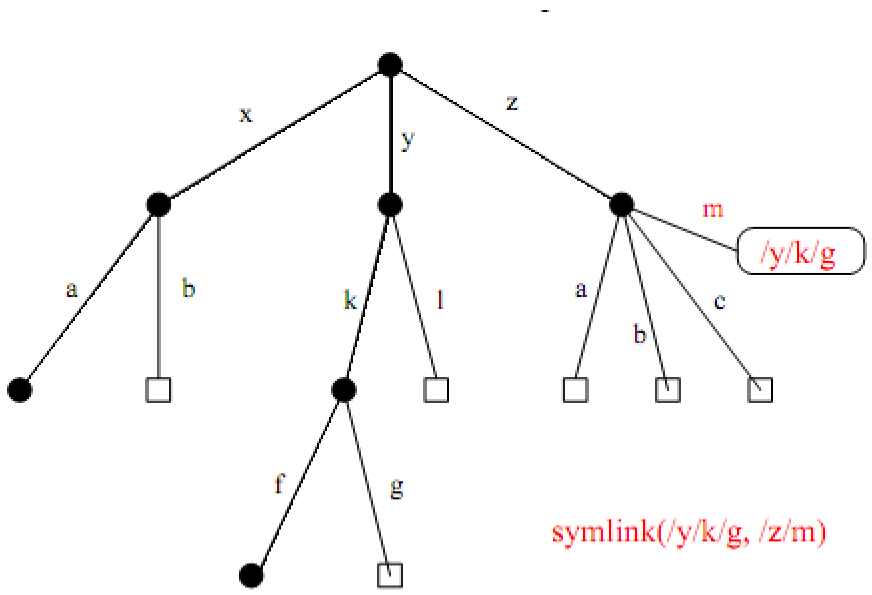
Hard Link Example



Symbolic Links

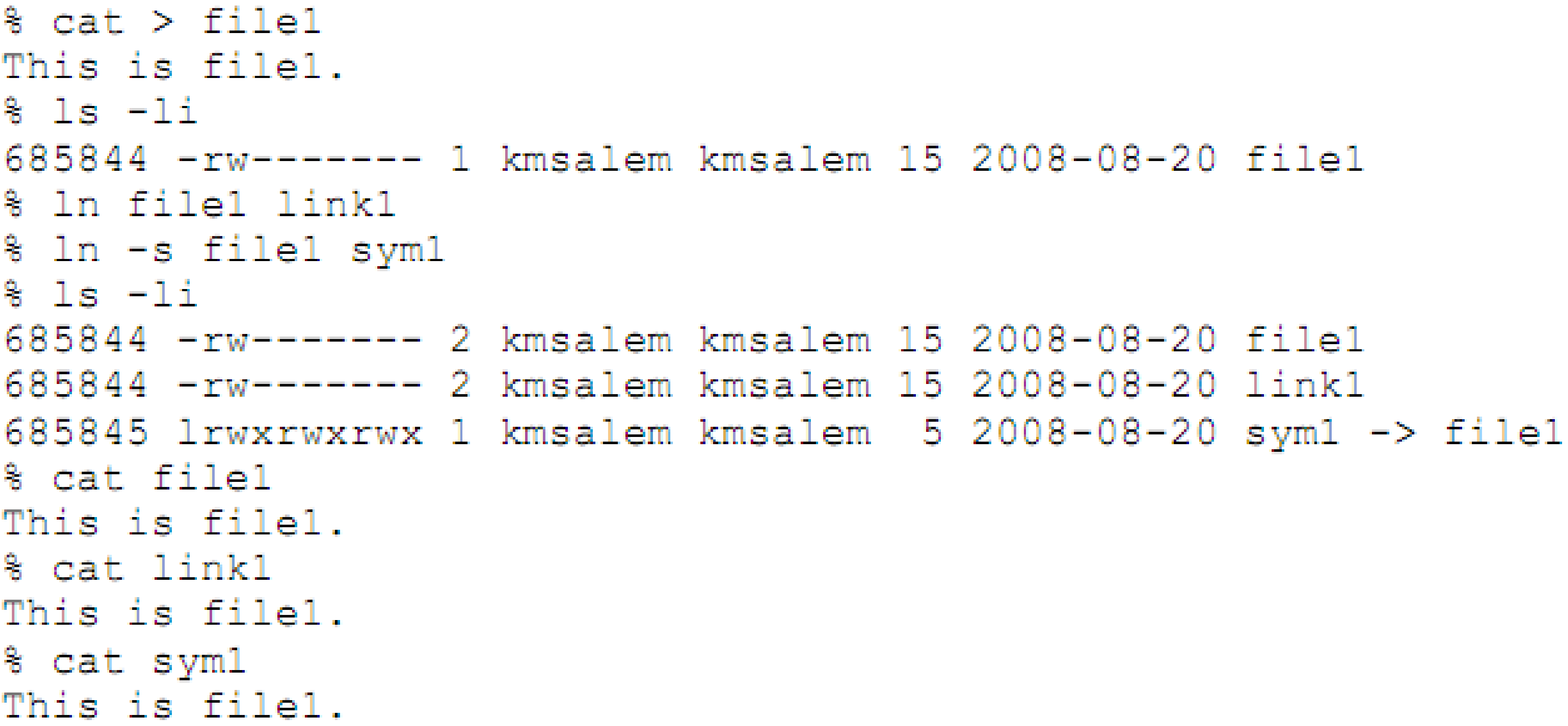
- a Symbolic link , or soft link, is an association between two names in the file namespace. Think of it is a way of defining a synonym for a filename.
  - symlink(oldpath,newpath) creates a symbolic link from newpath to oldpath, i.e., newpath becomes a synonym for oldpath.

Soft Link Example



symlink(/y/k/g, /z/m)

`/y/k/g` still has only one hard link after the `symlink` call. A new symlink object records the association between `/z/m` and `/y/k/g`. `open(/z/m)` will now have the same effect as `open(/y/k/g)`.



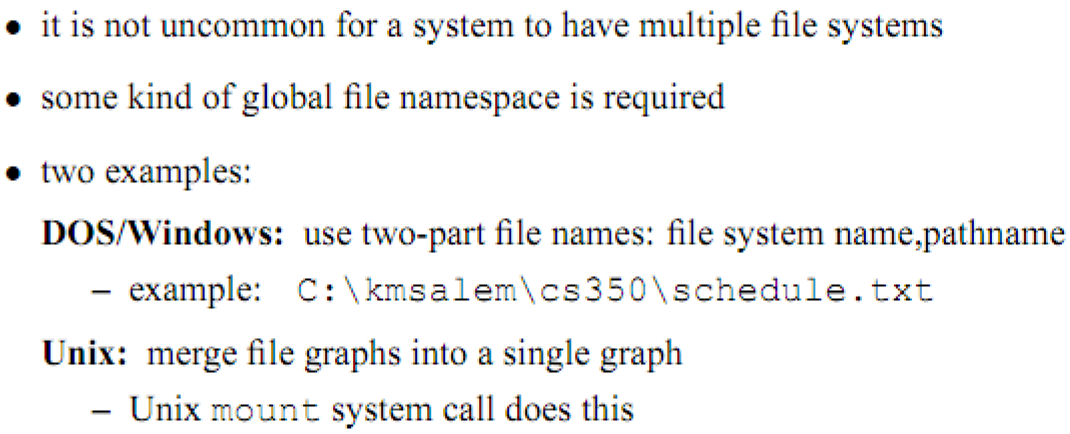
```
% cat > file1
This is file1.
% ls -li
685844 -rw------- 1 kmsalem kmsalem 15 2008-08-20 file1
% ln file1 link1
% ln -s file1 sym1
% ls -li
685844 -rw------- 2 kmsalem kmsalem 15 2008-08-20 file1
685844 -rw------- 2 kmsalem kmsalem 15 2008-08-20 link1
685845 lrwxrwxrwx 1 kmsalem kmsalem  5 2008-08-20 sym1 -> file1
% cat file1
This is file1.
% cat link1
This is file1.
% cat sym1
This is file1.
```

A file, a hard link, a soft link.

```
% /bin/rm file1
% ls -li
685844 -rw------- 1 kmsalem kmsalem 15 2008-08-20 link1
685845 lrwxrwxrwx 1 kmsalem kmsalem  5 2008-08-20 sym1 -> file1
% cat link1
This is file1.
% cat sym1
cat: sym1: No such file or directory
% cat > file1
This is a brand new file1.
% ls -li
685846 -rw------- 1 kmsalem kmsalem 27 2008-08-20 file1
685844 -rw------- 1 kmsalem kmsalem 15 2008-08-20 link1
685845 lrwxrwxrwx 1 kmsalem kmsalem  5 2008-08-20 sym1 -> file1
% cat link1
This is file1.
% cat sym1
This is a brand new file1.
```

Different behaviour for hard links and soft links.

Multiple File Systems



- it is not uncommon for a system to have multiple file systems
- some kind of global file namespace is required
- two examples:

  DOS/Windows: use two-part file names: file system name,pathname

  – example: `C:\kmsalem\cs350\schedule.txt`

  Unix: merge file graphs into a single graph

  – Unix `mount` system call does this