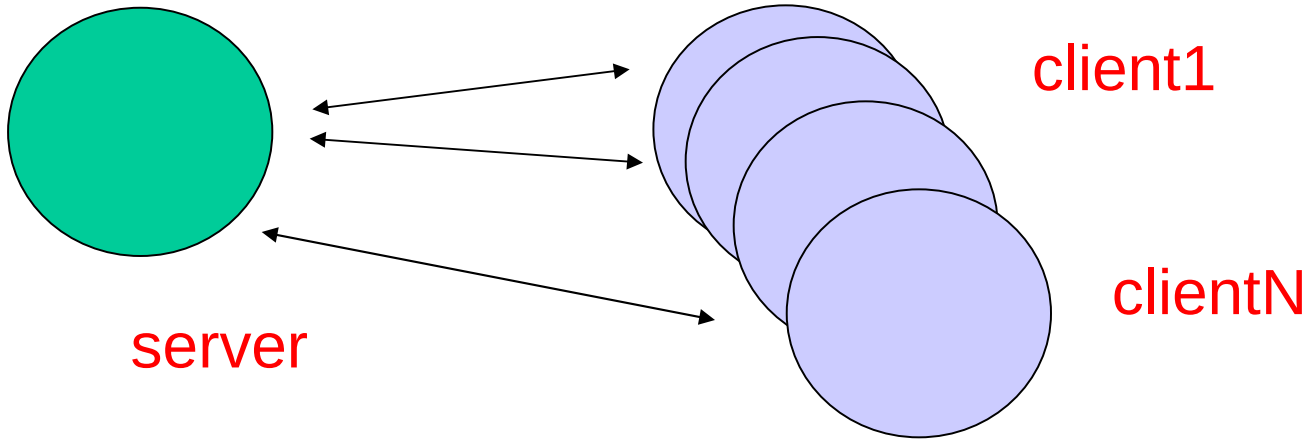
`sfat`

Progetto LCS 06-07

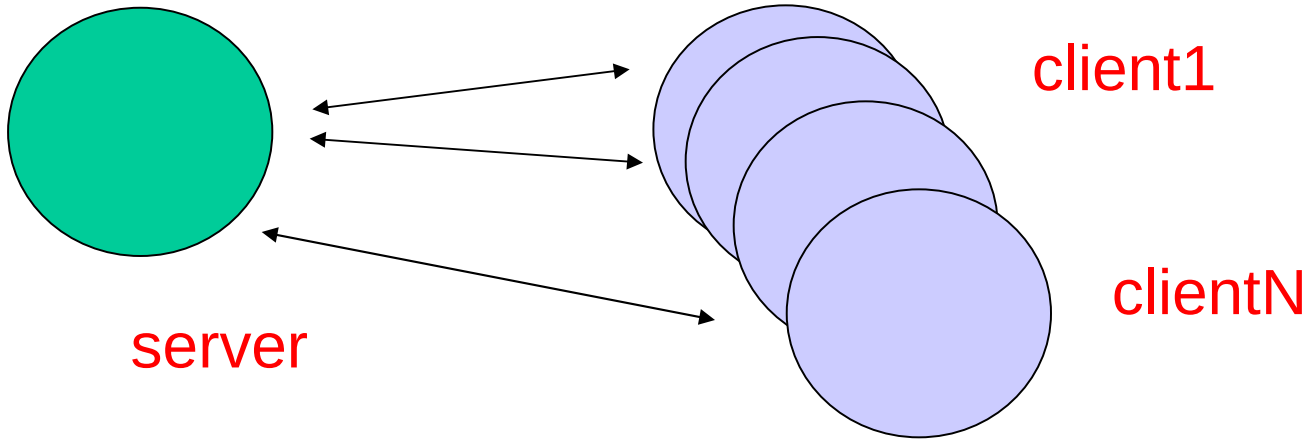
a Simplified FAT filesystem

Struttura complessiva



- Processo server
  - gestisce un filesystem FAT semplificato
  - il *device* che ospita il filesystem è un normale file unix
  - le richieste di operazioni sono inviate tramite socket
  - il server è multithreaded

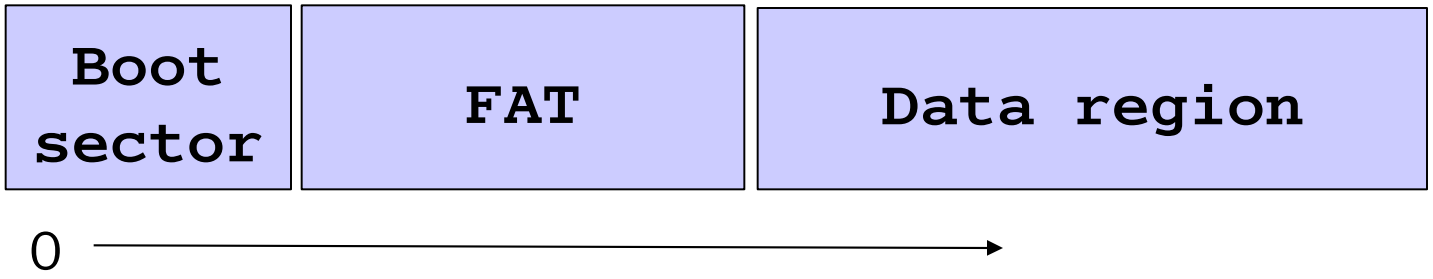
Struttura complessiva (2)



- Client
  - sono comandi unix che realizzano alcune operazioni sul file system
    - **`mkdir, ls, cp etc …`**
  - ogni comando crea una connessione con il client su un socket AF_UNIX

Il file *device*

- È organizzato in tre parti:
  - *boot sector*, *file allocation table* (FAT) e *data region*



Formato del filesystem

- Informazioni del boot sector
  - identificatore del file system (F)
  - numero di blocchi disponibili nella data region
  - dimensione dei blocchi, può essere:
    - 128, 1024, 2048, 4096
- FAT
  - normale file allocation table, vedi corso di SO
  - **`0x00000000`** (empty) , **`0xFFFFFFFF`** (End File)
- Data region
  - tutti i blocchi utilizzabili per memorizzare file o directory

Formato del filesystem (2)

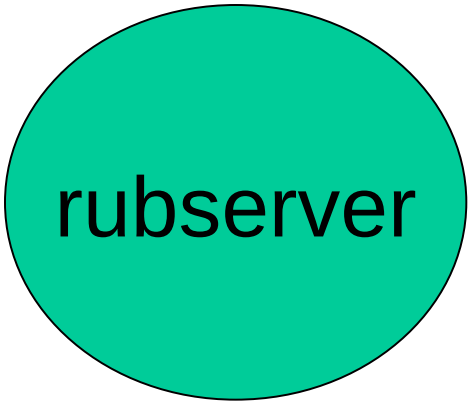
- Formato directory
  - ogni directory corrisponde ad una *directory table*, memorizzata come un normale file sfat
  - nella directory table c'è una enrty per ogni directory/file contenuto, più quelle per "." e ".."
- Directory entry
  - **`used`**: entry in uso o no
  - **`Name`**: nome del file/directory
  - **`Attribute`**: file o directory
  - **`Index`**: indirizzo primo blocco
  - **`Len`**: lunghezza in byte del file

Formato del filesystem (3)

- Root directory (/)
  - si trova nel blocco 0 della *data region*
- Formattazione di un nuovo FS
  - viene effettuata usando il comando

    `sfat_create dname nblk size`
  - **`dname`**: nome del nuovo filesystem device
  - **`nblk`**: numero dei blocchi nella data region
  - **`size`**: ampiezza di ogni blocco

Come funziona il server



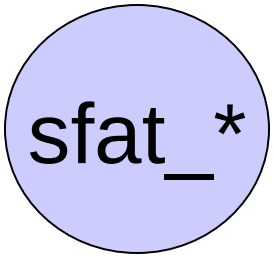
`fsfile`

- Attivazione server:

  `bash:~$ server fsfile`

  attiva un processo server che
  - effettua il mounting del file system da **`fsfile`**, caricando in memoria le informazioni necessarie
  - si mette in attesa di richieste dai client

Comandi client

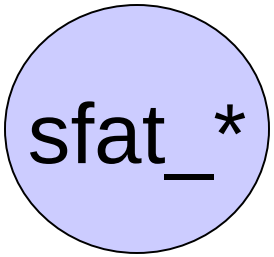



```
sfat_mkdir
sfat_ls
sfat_mkfile
sfat_append
sfat_read
sfat_cp
```

Usano esclusivamente path name assoluti

```
/dir1/dir2/dir3
/dir1/./../dir2/dir3
```

Comandi client (2)

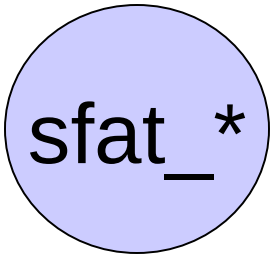



```
sfat_mkdir
sfat_ls
sfat_mkfile
sfat_append
sfat_read
sfat_cp
```

```
bash:~$ sfat_mkdir /dir1/dir2/dir3
sfat_mkdir: /dir1/dir2/dir3 created!
bash:~$
```

- crea **dir3** in **/dir1/dir2/**
  - la directory **/dir1/dir2/** deve esistere

Comandi client (3)

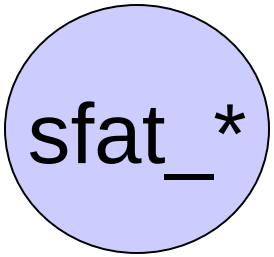



```
sfat_mkdir
sfat_ls
sfat_mkfile
sfat_append
sfat_read
sfat_cp
```

```
bash:~$ sfat_ls /dir1/dir2
sfat_ls: listing /dir1/dir2
.       ..      /dir3
bash:~$
```

– effettua il listing del contenuto (TAB separato)

Comandi client (4)

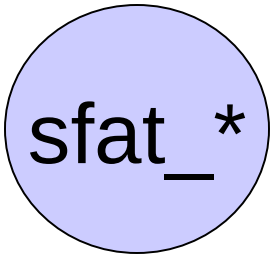



```
sfat_mkdir
sfat_ls
sfat_mkfile
sfat_append
sfat_read
sfat_cp
```

```
bash:~$ sfat_mkfile /dir1/dir2/f
sfat_mkfile: /dir1/dir2/f created!
bash:~$
```

– crea il file **`f`** in **`/dir1/dir2/`**
  - la directory **`/dir1/dir2/`** deve esistere

Comandi client (5)

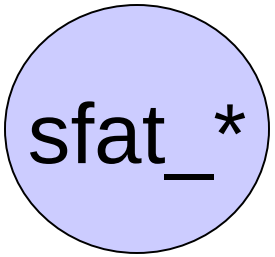



```
sfat_mkdir
sfat_ls
sfat_mkfile
sfat_append
sfat_read
sfat_cp
```

```
bash:~$ sfat_append /dir1/f “stringa”
sfat_append: appended “stringa”
bash:~$
```

– aggiunge **`string`** in coda al file **`/dir1/f`**
  - la directory **`/dir1/`** deve esistere
  - il terminatore non viene copiato

Comandi client (6)

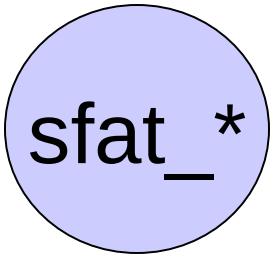



```
sfat_mkdir
sfat_ls
sfat_mkfile
sfat_append
sfat_read
sfat_cp
```

```
bash:~$ sfat_read /dir1/f 0 200
sfat_read: read /dir1/f “stringa”
bash:~$
```

– legge nel file **`/dir1/f`** a partire da offset 0 al più 200 caratteri

Comandi client (7)

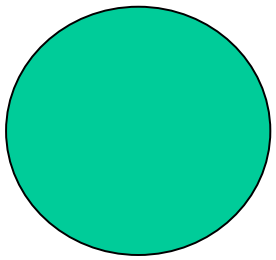



```
sfat_mkdir
sfat_ls
sfat_mkfile
sfat_append
sfat_read
sfat_cp
```

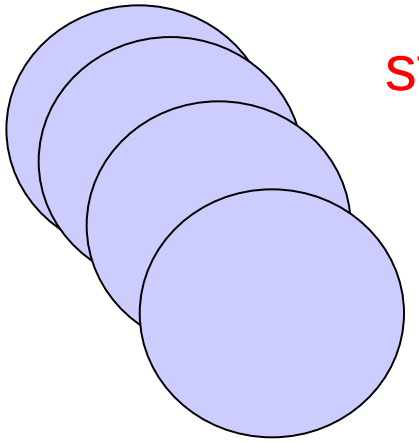
```
bash:~$ sfat_cp /dir1/f /dir1/g
sfat_cp: /dir1/g written
bash:~$
```

- crea il file **`/dir1/g`** e ci copia il contenuto di **`/dir1/f`**
- **`/dir1/g`** NON deve esistere

Cosa deve essere realizzato



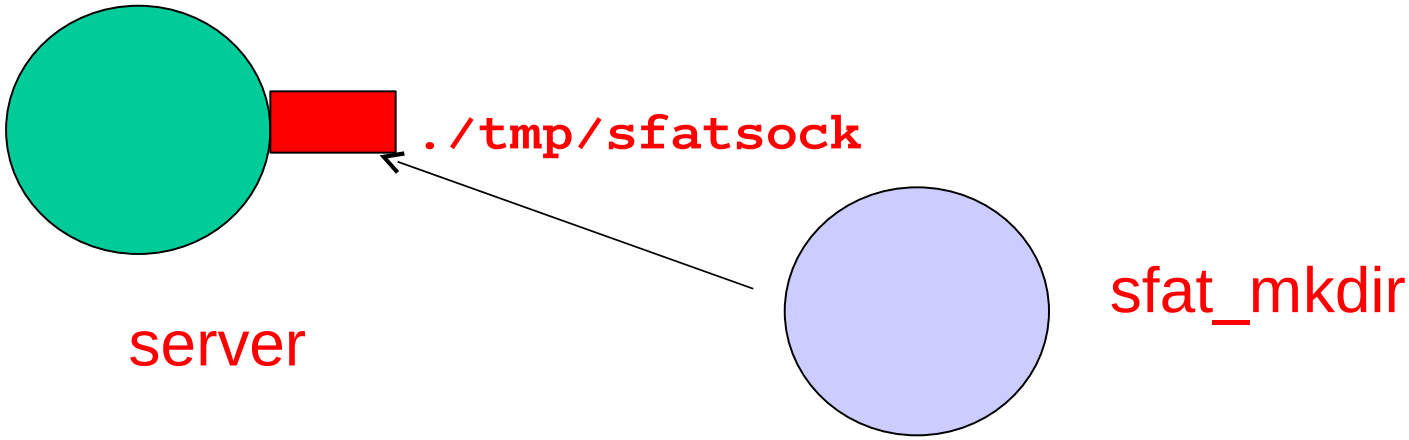
server



sfat_*

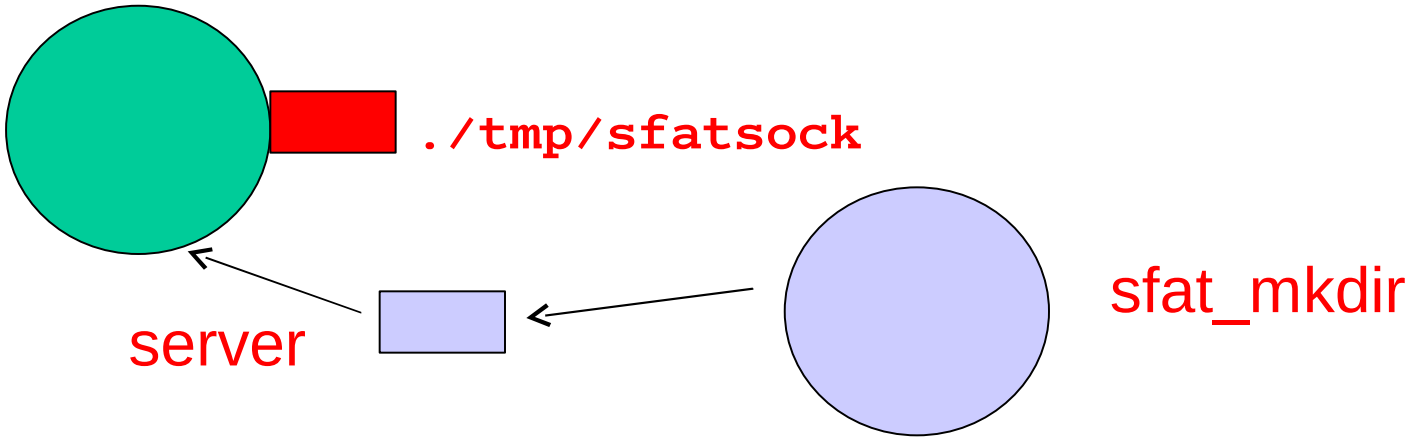
- **`libfat`** libreria di operazioni base sul filesystem (usata dal server)
- **`lcscom`** libreria di operazioni base sui socket
- **`server`** multithreaded
- comando **`sfat_create`**
- comandi client **`sfat_*`**

Interazioni client server



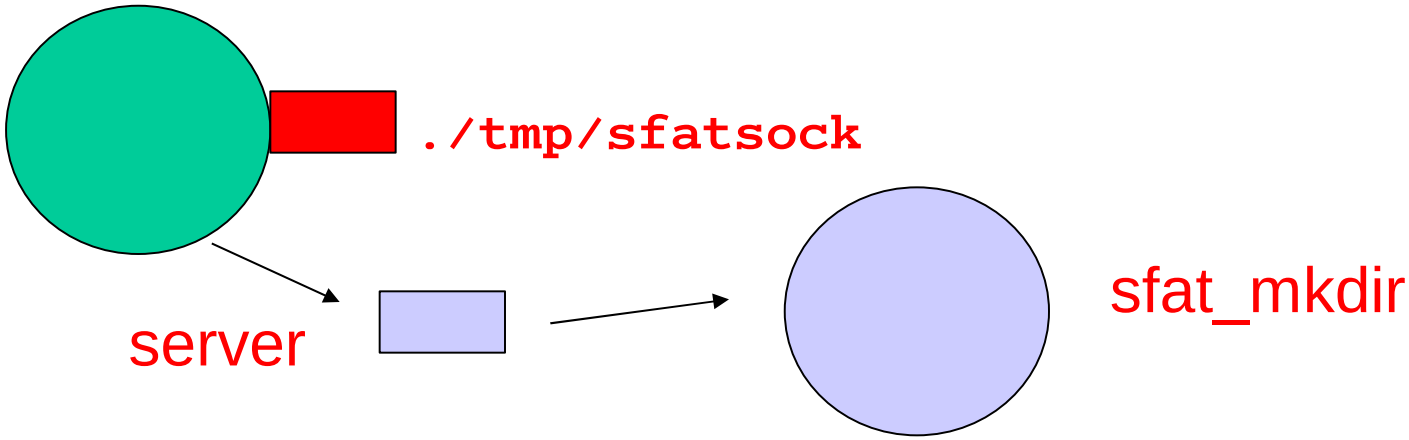
- I client si connettono al server attraverso un socket AF_UNIX dal nome fissato

Interazioni client server (2)



- Cinque richieste possibili dal formato fissato nelle specifiche
  - **MSG_MKDIR**, **MSG_LS**, **MSG_FREAD**, **MSG_FWRITE**, **MSG_MKFILE**,

Interazioni client server (3)



- 2 risposte possibili dal formato fissato nelle specifiche
  - **`MSG_OK`**, **`MSG_ERR`**,

Specifiche

- **`sfat0607.pdf`**
  - fornisce la visione globale della struttura del progetto
- **`libfat lcscom`**
  - la specifica delle strutture dati e delle funzioni da realizzare è nei .h del kit (`make doc` per ottenere la versione doxygen)
- testing finale
  - per libfat sono forniti dei test unitari che usano **`cunit`**
  - per server e client sono forniti 2 script di test

Alcune info

- Scadenze

  Per il bonus : 20/07/2007

  Finale : 1/02/2008

- è possibile consegnare e fare l'orale a partire da oggi
  - mediamente passa 1 settimana fra consegna e orale
- per fare l'orale bisogna aver svolto anche i primi due frammenti

Come e cosa si deve consegnare

- Esclusivamente usando gli script forniti nel materiale del kit
- Attenzione alla Relazione
  - le specifiche spiegano cosa bisogna metterci