My MailBox (mmbox): specifiche per il progetto d'esame Ver. 2.0 (specifiche estese)

Sistemi Operativi, Modulo 2
Anno Accademico 2010-2011

Claudio Cilli

Igor Melatti - Angelo Spognardi

Obiettivo

- Questo modulo descrive le **specifiche estese del progetto** da svolgere per l'esame di Sistemi Operativi, modulo 2 - AA 2010-2011.
-

Il progetto richiede lo sviluppo di un semplice gestore di posta, denominato `mmbox` (My MailBox)

NOTA BENE:

- Il progetto deve costituire **una creazione originale**, quindi non è possibile condividere parti di codice o di relazione con altri studenti/gruppi, o copiare contenuti derivanti da altre fonti.
- Il progetto puo' essere svolto individualmente o in gruppo (**max 3 persone**): i gruppi composti da 2 oppure da 1 persona potranno ottenere un **bonus** da aggiungere alla valutazione dell'esame.
- Per gli appelli di Giugno/Luglio e Settembre 2011 si richiede il soddisfacimento delle sole **specifiche di base**, mentre per quelli successivi si applicano le **specifiche estese**.

Specifiche - I

Il software package My MailBox `mmbox`:

- dovrà realizzare un **sistema di posta elettronica locale** con:
  - **Gestione degli utenti**
  - **Gestione delle quote**
  - **Scambio di messaggi e allegati**
- dovrà essere implementato come un'**applicazione di tipo server** di nome **`mmboxd`,** consistente in un eseguibile Linux (cioè NON come modulo kernel) **che gestisce una directory di mailbox**, denominata **`${HOME}/.mmbox`,** contenente **un file di messaggi per ogni utente del sistema**
- dovrà fornire un'**interfaccia interattiva di gestione del sistema**, denominata **`mmboxman`,** per **l'avvio/terminazione di `mmboxd`**, per la gestione degli utenti e delle quote e per **il monitoraggio a *run-time* del sistema,** con l'accesso alle statistiche sui messaggi scambiati e la situazione delle quote
- dovrà inoltre fornire una libreria statica **`libmmboxd.a`** che permetterà di scrivere applicazioni capaci di interagire con il server **`mmboxd`**

Specifiche – II

- la libreria dovrà fornire alle applicazioni client **un'API** la richiesta di invio, lettura e gestione dei messaggi con **gestione dello stato di una connessione** (vedi autenticazione/login)
- dovrà implementare un meccanismo basato sullo **scambio di dati via FIFO** tra le applicazioni client e il programma server, che supporti le richieste di invio, ricezione e consultazione dei messaggi
- dovrà implementare un **meccanismo di protezione** delle mailbox gestito da **`mmboxd`** per impedire che un qualsiasi utente possa accedere ai messaggi di un altro utente

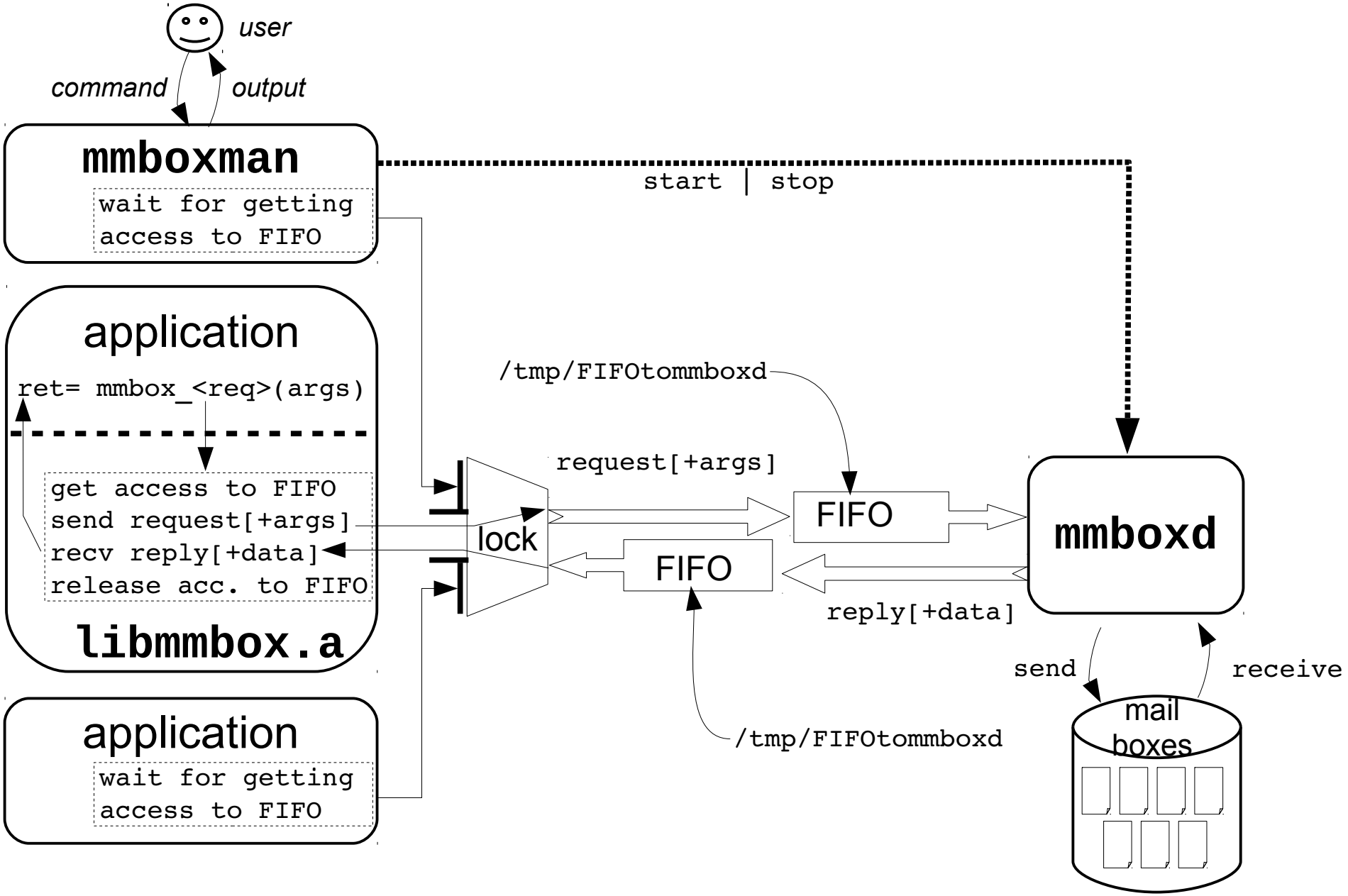
Specifiche implementative

- il server **`mmboxd`** dovrà **ricevere, memorizzare, organizzare e mantenere integre** le informazioni relative allo stato delle mailbox degli utenti, mediante opportune strutture di dati
- lo scambio di messaggi tra applicazioni e server **`mmboxd`** deve avvenire tramite **una coppia di FIFO** per la comunicazione full-duplex:
  `/tmp/FIFOtommboxd` : richieste da applicazione a **`mmboxd`**
  `/tmp/FIFOfrommboxd` : risposte da **`mmboxd`** ad applicazione
- il server **`mmboxd`** alla partenza deve ripristinare (o creare se appena installato) lo stato del sistema My Mailbox, recuperando la lista degli utenti e la situazione delle mailbox e infine attendere le richieste da parte delle applicazioni
- il server **`mmboxd`**, appena giunge una richiesta, la analizza e, se valida, esegue l'operazione richiesta, ritornando SEMPRE un messaggio al richiedente, che comunica il codice di ritorno (successo/errore) ed eventuali dati di output

Specifiche implementative

- la libreria **`libmmbox.a`** dovrà implementare all'interno delle sue funzioni un **meccanismo di lock esclusivo** delle FIFO per garantire l'atomicità della procedura *<richiesta operazione-ricezione risposta>*
- la terminazione del server **`mmboxd`** deve avvenire a seguito dell'**invio del segnale `SIGTERM`**, che attiva il cleanup delle strutture dati e procedure in corso e poi l'exit
- tutti i file necessari al sistema devono essere sistemati nella hiddeb directory **`${HOME}/.mmbox`** e devono essere protetti dalla lettura/scrittura di utenti non autorizzati
- al primo avvio del server **`mmboxd`**, l'utente *admin* viene creato di default e la password viene impostata interattivamente
-
- NOTA: dovrà essere possibile avviare **`mmboxd`** anche direttamente da riga di comando (ovvero anche senza usare **`mmboxman`**)

Schema architetturale



Funzioni da implementare in `libmmbox.a` – I

Nota:

- tutti i tipi NON definiti dall'ANSI C (typedef tipo `size_t, pid_t...`) devono essere identici a quelli definiti per Linux, quindi ci si deve riferire agli opportuni include file.
- Per ogni comando ci possono essere delle situazioni di errore; quelle fornite nel seguito sono esemplificative e **non esaustive** (altre condizioni d'errore sono possibili e devono essere gestite)

Funzioni di accesso e autenticazione

- `int mmbox_connect(char *username);`
  – richiede l'accesso ai servizi del server: deve essere il primo comando di ogni utente (client). Ritorna un valore < 0 in caso di errore (ad es. utente inesistente) o >= 0 in caso di successo; il valore ritornato deve poi essere usato per `mmbox_login`.
- `int mmbox_login(int token, char *password);`
  – richiede l'accesso ai servizi del server: deve essere invocato dopo che `mmbox_connect` sia stato eseguito con successo. Il parametro `token` deve essere uguale al valore tornato da `mmbox_connect`. Ritorna un valore != 0 in caso di errore (ad es. password sbagliata) o 0 in caso di successo
- `int mmbox_quit();`
  – conclude la sessione (per aver ancora accesso ai servizi, occorre eseguire nuovamente `mmbox_connect`). Ritorna un valore != 0 in caso di errore (ad es. nessuna connessione attiva) o 0 in caso di successo

Funzioni da implementare in `libmmbox.a` – II

Funzioni di gestione messaggi (statistiche)

- `int mmbox_stat(struct mmbox_stat_t *result);`
  – ritorna la situazione della mailbox riempiendo la seguente struttura:

```
struct mmbox_stat_t {
  int used_space; /* spazio usato */
  int free_space; /* spazio libero rispetto alla quota */
  char *user; /* utente attualmente loggato */
  int num_msg; /* numero di messaggi (anche marcati) */
};
```

  – ritorna un valore != 0 in caso di errore (ad es. destinatario sconosciuto) o 0 in caso di successo
- `int mmbox_list(struct mmbox_mail **l, int *num_msg);`
  – ritorna la situazione della mailbox in un vettore `l` di `num_msg` elementi:

```
struct mmbox_mail {
  char *sender, *recipient; /* mittente e destinatario */
  char *obj, *date; /* oggetto e data del messaggio */
  char flags; /*letto o no,marcato o no per la cancellazione*/
  size_t size; /*dimensione in bytes del corpo del messaggio*/
};
```

  – ritorna un valore != 0 in caso di errore (ad es. non connesso) o 0 in caso di successo

Funzioni da implementare in `libmmbox.a` – III

Funzioni di gestione messaggi (invio/ricezione)

- `int mmbox_send(char *dest, char *obj, void *buf, size_t size);`
  – invia un messaggio. Prende in input rispettivamente: il destinatario, l'oggetto del messaggio, il corpo del messaggio, la dimensione in bytes del corpo del messaggio. Ritorna un valore != 0 in caso di errore (ad es. destinatario sconosciuto, spazio nella mailbox esaurito...) o 0 in caso di successo
- `int mmbox_rcv(int id, void *buf, size_t size);`
  – restituisce in `buf` il corpo del messaggio (di dimensione `size`) di indice `id` nella lista di messaggi. Ritorna un valore != 0 in caso di errore (ad es. `id` sconosciuto) o 0 in caso di successo
- `int mmbox_delete(int id);`
  – marca per la cancellazione il messaggio di indice `id` nella lista di messaggi. Ritorna un valore != 0 in caso di errore (ad es. `id` sconosciuto) o 0 in caso di successo. L'effettiva eliminazione avviene alla chiusura della connessione (in seguito alla `mmbox_quit()`).
- `int mmbox_resume(int id);`
  – toglie il mark per la cancellazione al messaggio di indice `id` nella lista di messaggi. Ritorna un valore != 0 in caso di errore (ad es. `id` non marcato) o 0 in caso di successo

Funzioni da implementare in `libmmbox.a` – IV

Funzioni di gestione allegati (invio/ricezione)

- `int mmbox_send_attach(char *dest, char *obj, void *buf, size_t size, char *filename);`
  – Come la `mmbox_send`, ma invia anche il file il cui path è `filename`. Ritorna un valore != 0 in caso di errore (come la `mmbox_send`, più ad es. il path `filename` non e' valido) o 0 in caso di successo
- `int mmbox_rcv_attach(int id, char *dest);`
  – Copia nel file il cui path è `dest` l'attachment del messaggio `id`. Ritorna un valore != 0 in caso di errore (ad es. uno degli il messaggio non ha attachment) o 0 in caso di successo

Programma di gestione delle mailbox `mmboxman` - I

- il programma **`mmboxman`** presenta un prompt all'utente e, iterativamente, rimane in attesa dell'inserimento di un comando, il quale viene validato e poi eseguito
- l'output risultante viene mostrato a video
- un comando non valido deve generare un messaggio di errore

- appena avviato, mmboxman richiede l'inserimento della password di *admin*, poiché soltanto questi è abilitato all'uso del manager

Comandi da implementare in `mmboxman` - I

- **`start`** - avvia il server **`mmboxd`** e stampa messaggio di successo/errore.
- **`stop`** – termina il server **`mmboxd`** e stampa messaggio di successo/errore.
- **`exit`** - esce da mmboxman.
- **`stat [USERNAME]`** - stampa la lista degli utenti, con relative quote. Se è specificato username, stampa anche spazio libero e spazio occupato nella relativa mailbox.
- **`adduser USERNAME`** - aggiunge un utente al sistema con la quota di default.
- **`passwd USERNAME PASSWORD`** - imposta la password per un utente.
- **`deluser USERNAME`** - rimuove un utente dal sistema con la relativa mailbox.
- **`qmod USERNAME NEWSIZE`** - modifica la quota dell'utente username. Se la nuova dimensione è più piccola dello spazio occupato dai messaggi, la riduzione deve avvenire non appena l'utente avrà svuotato opportunamente la mailbox.

Comandi da implementare in `mmboxman` - II

- **`list`** – richiede la lista dei messaggi nella mailbox di admin (ogni messaggio ha il suo ID)
- **`retrieve ID`** – richiede la stampa del messaggio ID
- **`get_attach ID path`** – richiede di salvare in PATH l'attachment del messaggio ID
- **`delete ID`** – segna il messaggio ID come “da eliminare”
- **`resume ID`** – elimina lo stato “da eliminare” dal messaggio ID
- **`compose USERNAME [PATH]`** - inizia la composizione di un messaggio per l'utente username. La prima riga letta è l'oggetto del messaggio, quella successiva il testo del messaggio. Il secondo argomento è opzionale: se presente, va interpretato come il path del file da inviare in attachment.

Package da consegnare - I

- La compilazione degli eseguibili e librerie del package **`mmbox`** deve essere gestita mediante **`Makefile`**
- L'installazione degli eseguibili **`mmboxd, mmboxman`** e libreria **`libmmbox.a`** deve avvenire nella directory **`$HOME/.mmbox`** e **NON** deve richiedere il privilegio di **`root`**
- Il progetto deve essere documentato mediante una relazione che descrive:
  - l'organizzazione del package: albero delle directory, identificazione e caratterizzazione dei file sorgenti (.c) e include (.h) (funzionalita' assegnate ai vari sorgenti, definizione e ripartizione dei dati globali, tipi definiti di dati, ...)
  - le considerazioni per l'installazione e il test con eventuali esempi di comandi da effettuare
  - le eventuali funzionalita' aggiuntive opzionali implementate
  - la struttura dei programmi **`mmboxd`** e **`mmboxman`** con il dettaglio delle principali scelte implementative (strutture di dati, algoritmi, flussi di controllo a livello macro, ...)
  - la struttura delle funzioni della libreria **`libmmbox.a`**
  - il protocollo di comunicazione via FIFO (struttura e tipo dei messaggi di richiesta e di risposta)
  - il meccanismo di controllo per l'accesso esclusivo alle FIFO
  - la gestione degli errori con la lista dei codici di errore definiti
  - le eventuali limitazioni dovute all'implementazione

Package da consegnare - II

- La consegna deve consistere in un package, da spedire via mail, in formato tar compresso (tramite comando **`tar czf`**) col nome **`mmbox-2.0.tgz`**
- il file **`mmbox-2.0.tgz`** deve contenere una sola directory (con pathname relativo) di nome **`mmbox-2.0`**
- la directory **`mmbox-2.0`** deve contenere:
  - file **`AUTHORS`** (Cognome Nome Matricola, uno studente per linea)
  - file **`README`** (informazioni molto concise sulla compilazione, installazione e lancio del programma)
  - directory **`doc`** con relazione in formato PDF o OpenOffice
  - directory **`src`** (sorgenti .c), **`inc`** (include `.h`), **`test`** (file/programmi di test), **`bin`** (eseguibili generati), **`lib`** (librerie generate), **`obj`** (file compilati **`.o`**), ed eventuali altre directory
  - file **`Makefile`**, organizzato di modo che dare il comando `make` compili il tutto secondo necessità, `make install` esegua l'installazione in `${HOME}/.mmbox`, `make uninstall` cancelli l'installazione, `make clean` cancelli i file oggetto, `make cleanall` cancelli anche i file eseguibili (ma senza toccare i files nella directory `${HOME}/.mmbox`)