PROGETTO SISTEMI OPERATIVI – LINUX SYSTEM CALL 2018/2019

- **Prenotazione dello slot nel calendario esami-orali entro <u>domenica 9 Giugno 2019; (23:59)</u>**
- **Consegna elaborato entro <u>giovedì 20 Giugno 2019; (23:59)</u>**

Descrizione generale

Si vuole realizzare un'applicazione per la gestione dei seguenti servizi di sistema: Stampa, Salva, e Invio. La Figura 1 sottostante riporta lo schema dell'architettura Client-Server richiesta.

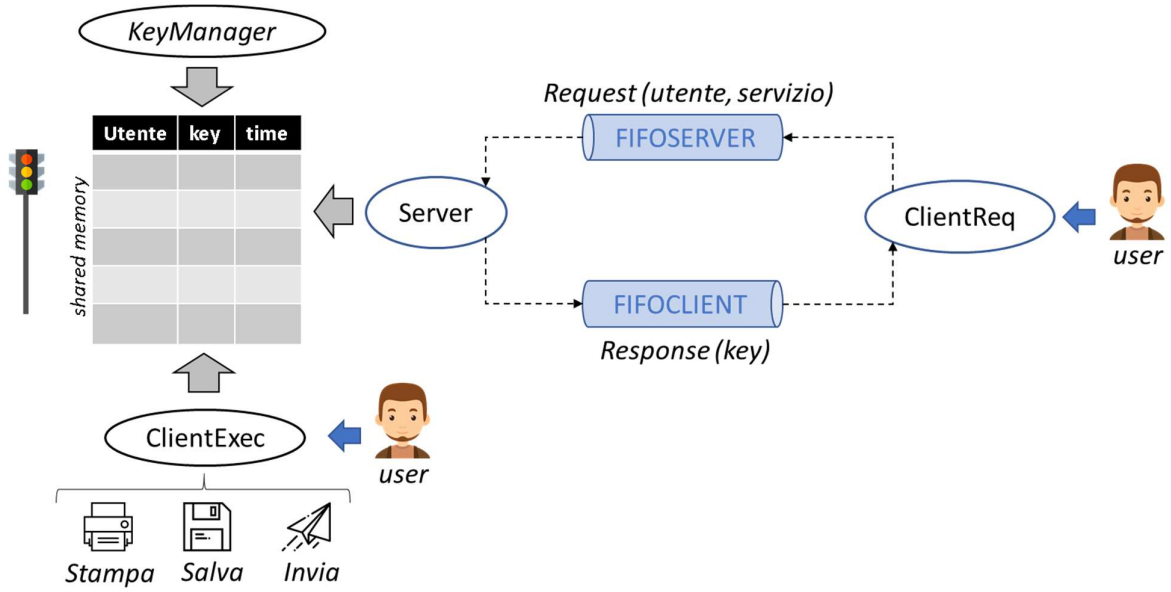

Figure 1: Architettura Client-Server

Si individui i seguenti programmi cooperanti in Figura 1:

- **ClientReq:** programma utilizzato dall'utente per richiedere una chiave di utilizzo per un servizio di sistema.
- **Server**: programma responsabile del rilasciato delle chiavi di utilizzo per i servizi di sistema disponibili.
- **KeyManager**: programma responsabile della memorizzazione e gestione di tutte le chiavi rilasciate per l'utilizzo di un servizio di sistema.
- **ClientExec**: programma incaricato di eseguire il servizio richiesto dall'utente.
- **Stampa:** programma che esplica il servizio di stampa.
- **Salva:** programma che esplica il servizio di salvataggio.
- **Invia**, programma che esplica il servizio di invio.

Di seguito viene riportato come i programmi sopra elencati, e l'utente, sono tenuti a cooperare nell'architettura Client-Server richiesta.

Cooperazione ClientReq e Utente.

Il programma ClientReq fornisce l'interfaccia da terminale che ogni utente utilizza per richiedere una chiave di utilizzo per un servizio di sistema.
All'avvio, ClientReq stampa un messaggio di benvenuto per l'utente, ed elenca i seguenti servizi di sistema: "Stampa", "Salva", e "Invia".
Successivamente, il programma richiede all'utente:

- codice identificativo dell'utente (stringa di caratteri)
- nome del servizio richiesto (stringa di caratteri)

ClientReq inoltra i dati forniti dall'utente al processo Server, ottiene una chiave di utilizzo numerica 'usa e getta' per il servizio richiesto, stampa la chiave ottenuta su terminale, ed infine termina. Esempio:

codice identificativo: "user1"
servizio: "Stampa"
chiave rilasciata del server: 1234

Cooperazione ClientReq e Server

La comunicazione tra ClientReq e Server avviene per mezzo di due FIFO.
FIFOSERVER è utilizzata dal programma ClientReq per richiede una chiave di utilizzo per un servizio. La richiesta deve essere un'istanza di un'opportuna struct denominata Request contente il codice identificativo dell'utente e il nome del servizio richiesto.
FIFOCLIENT è utilizzata dal programma Server per inviare al ClientReq una chiave numerica di utilizzo 'usa e getta' per il servizio richiesto. La risposta deve essere un'istanza di un'opportuna struct denominata Response.

Vincoli:

- ClientReq deve creare FIFOCLIENT prima di inviare i dati al server, e rimuovere FIFOCLIENT prima di terminare.
- Server deve creare FIFOSERVER all'avvio, e rimuovere FIFOSERVER prima di terminare.
- Se più ClientReq inviano al Server una Request, il Server deve gestisce sequenzialmente ogni richiesta.
- Server non deve fornire una chiave di utilizzo valida qualora l'utente richieda un servizio non disponibile.
- La chiave rilasciata dal server deve contenere in qualche forma la tipologia di servizio utilizzabile (implementazione libera e a carico dello studente).
- Una chiave non deve essere generata più di una volta.
- Server termina alla ricezione del segnale SIGTERM. Ogni altro segnale deve essere bloccato.

Cooperazione Server e KeyManager

Server e KeyManager condividono un segmento di memoria condivisa.
Ad ogni Request ricevuta, il processo Server memorizza nel segmento di memoria condivisa la seguente tripletta: (utente, chiave, timestamp).
Ad un intervallo temporale fissato di 30 secondi, il processo KeyManager rimuove dal segmento di memoria condivisa tutte le chiavi con un timestamp più vecchio di 5 minuti rispetto all'ora corrente di sistema.

Vincoli:

- Server deve creare il segmento di memoria condivisa all'avvio, e rimuovere tale segmento prima di terminare.
- KeyManager è un processo figlio del processo Server.
- Server, KeyManager e ClientExec (vedi sotto) accedono in modo mutualmente esclusivo al segmento di memoria condiviso.
- KeyManager termina alla ricezione del segnale SIGTERM.
- Il processo Server invia il segnale SIGTERM al processo KeyManager dopo aver ricevuto il segnale SIGTERM dall'utente.

Cooperazione ClientExec e Utente.

Il programma ClientExec è il programma utilizzato dall'utente per usufruire del servizio di sistema richiesto. ClientExec riceve da riga di comando i seguenti argomenti:

1. codice identificativo dell'utente
2. chiave rilasciata dal server
3. lista argomenti da riga di comando per il servizio

Prima di eseguire il servizio richiesto dall'utente, ClientExec accede al segmento di memoria condiviso tra Server e KeyManager per verificare la validità della chiave fornita.
Una chiave non è valida se:

- la coppia (utente, chiave) non è presente del segmento di memoria condivisa.
- La chiave di utilizzo è già stata utilizzata.

Se la chiave fornita è valida, la chiave viene cancellata dal segmento di memoria condiviso. Successivamente, ClientExec esegue il servizio richiesto utilizzando la lista dei parametri da riga di comando inizialmente fornita.

Vincoli:

- Server, KeyManager e ClientExec accedono in modo mutualmente esclusivo al segmento di memoria condiviso.
- L'esecuzione del servizio richiesto deve avvenire per mezzo della system call exec.

Descrizione dei servizi forniti.
Ogni servizio fornito deve essere realizzato come programma eseguibile indipendente.
-Programma Stampa:
Il programma Stampa riporta su terminale tutti gli argomenti ricevuti da linea di comando.
-Programma Salva:
Il programma Salva riporta su file tutti gli argomenti ricevuti da linea di comando. Si assuma che il primo argomento sia il nome del file.
-Programma Invia:
Il programma Invia deposita tutti gli argomenti ricevuti da linea di comando in una coda di messaggi nella forma di un unico messaggio. Si assuma che il primo argomento sia la key identificativa di una coda di messaggi già presente nel sistema.

Tutto ciò non espressamente specificato nel testo del progetto è a scelta dello studente.

È vietato l'uso di funzioni C per la gestione del file system (*e.g.* fopen). Il progetto deve funzionare sui PC del laboratorio, rispettare il template fornito, ed essere compilabile con il comando make

Si ricorda che l'ammissione all'esame orale è possibile solo se rispettata la dati di consegna dell'elaborato.

Il presente elaborato permette un voto massimo di 25 trentesimi.
La consegna dell'esercitazione su MentOS/process-management, e del presente elaborato permette un voto massimo di 27 trentesimi.
La consegna dell'esercitazione su MentOS/process-management, MentOS/memory-management, e del presente elaborato permette un voto massimo di 30 e lode.

Consegna elaborato e-learning
Si richiede di rispettare la seguente struttura di cartelle per la consegna dell'elaborato:
/sistemi_operativi
/system-call/ (elaborato system call)
/MentOS/scheduler_algorithm.c
/MentOS/buddysystem.c
La directory sistemi_operativi deve essere compressa in un archivio di nome <matricola>_sistemi_operativi.tar.gz. L'archivio deve essere creato con il comando tar (no programmi esterni). L'archivio tar.gz deve essere caricato nell'apposita sezione sul sito di e-learning.