6h ONLINE Esercizi (laboratorio lez 12) Progr. Concorrente e Bash

Contenuti:

ESERCIZI SIMILI A QUELLI DELL'ESAME PROVA PRATICA 0b

Esercizio 1051ONLINE Programmazione concorrente ponte semplice

Esercizio 1052ONLINE Programmazione concorrente ponte complicato

Esercizio 1053 script bash

Esercizio 1054 script bash

Materiale per Prova Pratica 0NLINE

Download Materiale:

Scaricare il file con le **dispense** e gli **esempi** svolti a lezione

wget http://www.cs.unibo.it/~ghini/didattica/TREE4OS2021.tgz

Decomprimere l'archivio scaricato: tar xvzf TREE4OS2021.tgz

Viene creata una directory **TREE4OS2021** con dentro una sottodirectory **sistemioperativi** con dentro tutto il **materiale**.

Potete navigare tra il materiale con un normale browser aprendo l' URL

file://home/studente/VOSTRADIRECTORY/TREE4OS2021/sistemioperativi/dispenseSistOp2021.html

o anche

file://home/studente/VOSTRADIRECTORY/TREE4OS2021/sistemioperativi/index.html

Esercizio 1051ONLINE - **ponte pericolante semplice (1)**

Nell'ameno paesino di Villa Inferno, tra Cesena e Cervia, c'è una unica strada a due corsie fatta come un anello che gira tutto attorno al paese. Le auto girano continuamente in tondo, senza smettere mai. In un tratto della strada c'è un ponte pericolante e con una sola corsia, così **solo un' auto alla volta può attraversare il ponte**.

Ciascuna auto, quindi, **può attraversare solo quando nessuna altra auto è sul ponte**.
Ma bisogna anche rispettare un turno tra le auto del proprio lato. A tal scopo, su ciascun lato del ponte c'è un distributore di biglietti numerati crescenti che determinano l'ordine di attraversamento del ponte per le macchine che partono da quel lato. Ciascuna auto che vuole attraversare prende un biglietto sul proprio lato e attende il proprio turno tra quelle del proprio lato.

Per smaltire le code, **la regola di precedenza** stabilisce che **attraversa il ponte l'auto col biglietto più piccolo tra quelle che stanno sul lato in cui ci sono attualmente più auto in attesa di attraversare. In caso di parità, attraversa l'auto che gira in senso orario.**

Ciascuna auto impiega 1 secondo a percorrere il ponte e altri 5 secondi per percorrere il resto dell'anello. Ci sono 4 auto che viaggiano in senso orario e altre 4 in senso antiorario.
All'inizio le auto si trovano all'ingresso del ponte (ciascuna sul lato dipendente dal proprio verso di percorrenza).

la descrizione continua nella pagina seguente

Esercizio 1051ONLINE - **ponte pericolante semplice (2)**

Modellare ed implementare il sistema descritto, **utilizzando dei thread POSIX** per ciascuna figura (auto in senso orario e auto in senso antiorario) ed avvalendosi delle opportune strutture dati per la sincronizzazione.
Utilizzare come base dell'esercizio i seguenti files, aggiungendo il codice necessario:

pontepericolante_semplice.c DBGpthread.c DBGpthread.h printerror.h

Scrivere il Makefile per creare l'eseguibile. La mancanza del Makefile viene considerato un errore grave. Occorre inserire il controllo di errore nelle chiamate a funzione delle librerie dei pthread. In caso di errore grave, terminare il programma producendo un avviso a video.

Note implementative della base fornita:
ciascuna Auto è caratterizzata da:
- un senso di marcia, che può assumere valore **'O'** (ORARIO) oppure **'A'** (ANTIORARIO),
- e da un indice che va da 0 a (NUMAUTOORARIO-1) per le auto che viaggiano in senso orario, e da 0 a (NUMAUTOANTIORARIO-1) per le auto che viaggiano in senso antiorario..

I due distributori dei biglietti (stile fornaio) sono rappresentati da un vettore **bigliettoDistributore[2]** di due interi: in posizione 0 c'è il distributore per le auto orarie, in posizione 1 quello per le auto antiorarie.
Su ogni lato del ponte c'é un display che dice CHI E' IL PRIMO NELLA CODA SU QUEL LATO DEL PONTE. Ma non basta essere il primo della propria coda per attraversare, occorre rispettare le regole di precedenza descritte nella pagina precedente. I 2 display sono implementati con un vettore di due interi **biglietto[2]**
Nei vettori, l'indice 0 indica le entità usare dalle macchine che girano in senso ORARIO, l'indice 1 invece quelle per le macchine che girano in senso ANTIORARIO.

Esercizio 1052ONLINE - **ponte pericolante complicato (1)**

Nell'ameno paesino di Villa Inferno, tra Cesena e Cervia, c'è una unica strada a due corsie fatta come un anello che gira tutto attorno al paese. Le auto girano continuamente in tondo, senza smettere mai. In un tratto della strada c'è un ponte pericolante con una strettoia, così le auto possono attraversare solo in un senso di marcia per volta.

Un'auto può attraversare solo se è vera una delle due seguenti condizioni.

1) **Nessuna altra auto è sul ponte,**

oppure

2) **sul ponte c'è ancora un auto, ma questa ha già percorso almeno metà del ponte e va nello stesso verso dell'auto che vuole attraversare.**

Ma bisogna anche rispettare un turno tra le auto del proprio lato. A tal scopo, su ciascun lato del ponte c'è un distributore di biglietti numerati crescenti che determinano l'ordine di attraversamento del ponte per le macchine che partono da quel lato. Ciascuna auto che vuole attraversare prende un biglietto sul proprio lato e attende il proprio turno tra quelle del proprio lato.

Quando nessuna auto è sul ponte, la regola di precedenza stabilisce che può attraversare il ponte l'auto col biglietto più piccolo tra quelle che stanno sul lato in cui ci sono attualmente più auto in attesa di attraversare. In caso di parità, attraversa l'auto che gira in senso orario.

Ciascuna auto impiega 1 secondo a percorrere ciascuna delle metà del ponte e altri 10 secondi per percorrere il resto dell'anello. Ci sono 4 auto che viaggiano in senso orario e altre 4 in senso antiorario. All'inizio le auto sono all'ingresso del ponte (secondo il loro verso di percorrenza).

la descrizione continua nella pagina seguente

Esercizio 1052ONLINE - **ponte pericolante complicato (2)**

Modellare ed implementare il sistema descritto, **utilizzando dei thread POSIX** per ciascuna figura (auto in senso orario e auto in senso antiorario) ed avvalendosi delle opportune strutture dati per la sincronizzazione.
Utilizzare come base dell'esercizio i seguenti files, aggiungendo il codice necessario:

pontepericolante_complicato.c DBGpthread.c DBGpthread.h printerror.h

Scrivere il Makefile per creare l'eseguibile. La mancanza del Makefile viene considerato un errore grave. Occorre inserire il controllo di errore nelle chiamate a funzione delle librerie dei pthread. In caso di errore grave, terminare il programma producendo un avviso a video.

Note implementative della base fornita:
ciascuna Auto è caratterizzata da:
- un senso di marcia, che può assumere valore **'O'** (ORARIO) oppure **'A'** (ANTIORARIO),
- e da un indice che va da 0 a (NUMAUTOORARIO-1) per le auto che viaggiano in senso orario, e da 0 a (NUMAUTOANTIORARIO-1) per le auto che viaggiano in senso antiorario..

I due distributori dei biglietti (stile fornaio) sono rappresentati da un vettore **bigliettoDistributore[2]** di due interi: in posizione 0 c'è il distributore per le auto orarie, in posizione 1 quello per le auto antiorarie.
Su ogni lato del ponte c'é un display che dice CHI E' IL PRIMO NELLA CODA SU QUEL LATO DEL PONTE. Ma non basta essere il primo della propria coda per attraversare, occorre rispettare le regole di precedenza descritte nella pagina precedente. I 2 display sono implementati con un vettore di due interi **biglietto[2]**
Nei vettori, l'indice 0 indica le entità usare dalle macchine che girano in senso ORARIO, l'indice 1 invece quelle per le macchine che girano in senso ANTIORARIO.

Esercizio 1053 raggruppa.sh

Creare un file di testo cadutevic.txt come quello che segue:

1972	cesena	ribaltamento	leva_del_freno_infilata_in_una_coscia
1978	cesenatico	manubrio_strappato	fianco_grattugiato
1979	cesena	pedale_rotto	botta_agli_zebedei
1981	san_piero	drittone_in_un_campo	naso_rotto
1982	monte_cavallo	freni_allentati	niente
1983	monte_cavallo	freni_allentati	niente
2007	ciola_araldi	ribaltamento	incisivo_perso
2010	monte_cavallo	ghiaccio	niente

Ciascuna riga del file contiene 4 campi: l'anno di una caduta, la località della caduta, il motivo della caduta, i danni riportati.

Notare che quando un campo è formato da più di una parola le parole sono unite da un carattere underscore _ Ad esempio incisivo_perso

Notare che nessun campo motivo è una sottostringa di qualche altro campo motivo.

Realizzare uno script bash che emette sullo standard output alcune righe. In ciascuna riga compare un motivo della caduta e il numero delle volte che quella motivo è accaduto.

Potrebbe essere utile usare qualche file temporaneo in cui salvare informazioni parziali.

Per evitare ripetizioni di righe in output si può usare un comando uniq che permette di eliminare le righe ripetute di un file. Usare il man per capire come funziona.

Esercizio 1054 stringhe_confinate.sh

Creare un file di testo cadutevic.txt come quello che segue:
Ciascuna riga del file contiene 4 campi: l'anno di una caduta, la località della caduta il motivo della caduta, i danni riportati.

```
"1972"  "cesena"          "ribaltamento"          "leva del freno  infilata  in  una  coscia"
"1978"  "cesenatico"      "manubrio  strappato"   "fianco  grattugiato"
"1979"  "cesena"          "pedale  rotto"         "botta  agli  zebedei"
"1981"  "san piero"       "drittone  in  un campo"  "naso  rotto"
"1982"  "monte  cavallo"  "freni  allentati"      "niente"
"1983"  "monte  cavallo"  "freni  allentati"      "niente"
"2007"  "ciola  araldi"   "ribaltamento"         "incisivo  perso"
"2010"  "monte  cavallo"  "ghiaccio"             "niente"
```

Notare che i singoli campi sono delimitati (prima e dopo) da un doppio apice " che fa parte del campo stesso.
Quindi la stringa "freno infilato in una coscia" viene considerato come un unico campo.

Realizzare uno script bash che scrive sullo standard output solo il TERZO campo di ciascuna riga del file cadutevic.txt

Soluzione Esercizio 1054 stringhe_confinate.sh
http://www.cs.unibo.it/~ghini/didattica/sistemioperativi/ESERCIZI/es1054_stringhe_confinate.tgz

SOLUZIONI

Soluzione Esercizio 1051 es1051_pontepericolante_semplice.c
http://www.cs.unibo.it/~ghini/didattica/sistemioperativi/ESERCIZI/es1051_pontepericolante_semplice.tgz

Soluzione Esercizio 1052 es1052_pontepericolante_complicato.c
http://www.cs.unibo.it/~ghini/didattica/sistemioperativi/ESERCIZI/es1052_pontepericolante_complicato.tgz

Soluzione Esercizio 1053 raggruppa.sh
http://www.cs.unibo.it/~ghini/didattica/sistemioperativi/ESERCIZI/es1053_raggruppa.tgz

```
#/bin/bash
FILEDIAPPOGGIO=temp.txt
if [[ -e ${FILEDIAPPOGGIO} ]] ; then
        rm -f ${FILEDIAPPOGGIO}
fi
while read DATA LUOGO MOTIVO DANNI ; do
        NUM=`grep ${MOTIVO} cadutevic.txt | wc -l`
        echo "${MOTIVO} ${NUM}" >> ${FILEDIAPPOGGIO}
done < cadutevic.txt
sort ${FILEDIAPPOGGIO} | uniq

rm -f ${FILEDIAPPOGGIO}
exit 0
```

Soluzione Esercizio 1054 stringhe_confinate.sh
http://www.cs.unibo.it/~ghini/didattica/sistemioperativi/ESERCIZI/es1054_stringhe_confinate.tgz