# Prova Pratica 017
# turno 1 gruppo 1

# 2017 febbraio 06

i file da consegnare **devono** essere collocati nella directory **CONSEGNA** dentro la home directory dell'utente studente.

# SUGGERIMENTO PRATICO PRELIMINARE
# PER CHI VUOLE FARE
# l' esercizio in cui servono i PROCESSI

Se avete in esecuzione tanti processi che hanno tutti nome

processo.exe

e li volete uccidere tutti,

potete killare tutti quei processi utilizzando il comando:

killall processo.exe

# Prova Pratica 017 - turno 1 gruppo 1

Download Materiale:

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

wget http://esameso.csr.unibo.it/TREE4OS1617.tgz

Decomprimere l'archivio scaricato: tar xvzf TREE4OS1617.tgz

Viene creata una directory **TREE4OS1617** 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/TREE4OS1617/sistemioperativi/dispenseSistOp1617.html**

**Esercizi d'esame: p**er chi ha difficoltà a superare la prova pratica, ho previsto due tipi di prove:

A. **una prova COMPLICATA**, e' la modalità normale che vi permette di raggiungere un **voto massimo** (nella prova pratica stessa) di **30Lode** ,
B. ed **una prova SEMPLICE**, un po' **meno complicata**, che però vi permette di raggiungere un **voto massimo di 24** perché l'esercizio di programmazione concorrente é meno difficile.

**Scegliete voi quale prova svolgere** in funzione della vostra preparazione.

La prova COMPLICATA è composta dagli esercizi **46 e 47**,

La prova SEMPLICE è composta dagli esercizi **45 e 47**.

Come vedere l'esercizio 47 è comune alle due prove.

Svolgete SOLO gli esercizi della prova che vi interessa.

I file da consegnare **devono** essere collocati nella directory **CONSEGNA** dentro la home directory dell'utente studente.

# Esercizio Esame Pratica**- 45- attacchini_con_timer** (semplice)

**Due** dipendenti comunali lavorano come **attacchini** e incollano ripetutamente avvisi di funerali in una bacheca cittadina. Tra gli strumenti hanno un **orologio**, uno solo per tutti e due.
I due attacchini operano come due entità separate, ma devono lavorare assieme. Quando un attacchino inizia a lavorare, deve aspettare che anche l'altro attacchino cominci a lavorare.
**Dal momento in cui entrambi hanno cominciato a lavorare**, l'unico orologio viene impostato per avvisare gli attacchini dopo che sono passati circa 5 secondi.
Al suono dell'orologio ciascun attacchino smette di lavorare e poi va via a prendere nuovi manifesti e colla, poi torna **immediatamente** e cerca di ricominciare a lavorare.

**Modellare ed implementare il sistema descritto**, utilizzando dei PROCESSI per ciascuna figura (due attacchini e l'orologio) ed avvalendosi delle opportune strutture dati per la sincronizzazione. Scrivere il Makefile per compilare e linkare i sorgenti. 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.

# Esercizio Esame Pratica-**46- attacchini_sincroni** (complicato)

**Due** dipendenti comunali lavorano come **attacchini** e incollano ripetutamente avvisi di funerali in una bacheca cittadina. **Quattro vecchietti** osservano ripetutamente i manifesti affissi.
I due attacchini operano come due entità separate, ma devono lavorare assieme. Quando un attacchino inizia a lavorare, deve aspettare che anche l'altro attacchino cominci a lavorare. **Dal momento in cui entrambi hanno cominciato a lavorare**, ciascun attacchino lavora per circa 2 secondi, poi smette di lavorare e infine va via a prendere nuovi manifesti e colla.
Ciascun attacchino tornerà alla bacheca dopo 2 secondi, cercando di ricominciare ad attaccare manifesti.
I 4 vecchietti, numerati con un indice da 0 a 3, possono guardare la bacheca tutti contemporaneamente. Ogni vecchietto guarda la bacheca per circa 3 secondi, poi va via e dopo altri (1+(indice%2)) secondi torna a guardare perché si è dimenticato cosa ha letto.
Quando gli attacchini si avvicinano alla bacheca e vorrebbero lavorare, i vecchietti che già stanno leggendo continuano a leggere e se ne vanno solo quando hanno finito di leggere, facendo aspettare gli attacchini. Invece, i vecchietti che non avevano ancora cominciato a leggere devono aspettare che gli attacchini abbiano finito di incollare e se ne siano andati.

**Modellare ed implementare il sistema descritto**, utilizzando dei thread POSIX per ciascuna figura (attacchino, vecchietto) ed avvalendosi delle opportune strutture dati per la sincronizzazione. Scrivere il Makefile per compilare e linkare i sorgenti. 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.

# Esercizio Esame Pratica **- 47 - ultimerighe.sh**

**Scrivere** uno script bash **ultimerighe.sh** che accetta un solo argomento a riga di comando. Questo unico argomento sara' il percorso, relativo o assoluto, per identificare univocamente un file esistente.
Se allo script viene passato un numero di argomenti diverso da 1, lo script deve mandare sullo **standard error** il messaggio "numero argomenti errato" e poi terminare restituendo come codice d'errore 1.
Se allo script viene passato esattamente 1 argomento, lo script deve controllare se il file specificato da quell'argomento esiste. Se il file non esiste, lo script deve mandare sullo **standard error** il messaggio "argomento non file" e poi terminare restituendo come codice d'errore 2.
Se invece quel **file** esiste, allora lo script deve far eseguire **in background** una sequenza di comandi , o di script, che:

prima aspetta 2 secondi, e poi
seleziona le ultime 3 righe del **file** e **aggiunge** quelle righe al file OUTPUT.txt nella directory in cui viene lanciato lo script.

Nel frattempo lo script sara' terminato restituendo 0.

Infine, scrivere uno script **chiama.sh** che esegue due volte lo script ultimerighe.sh, passandogli come argomento:
la prima volta il percorso di un file che esiste /usr/include/stdio.h
la seconda volta il percorso di un file che non esiste ./VICSCEMO.txt
La seconda chiamata serve a evidenziare se la gestione dell' errore funziona correttamente.
Assumiamo che nel percorso del file non compaiano spazi bianchi.

# Esercizio Esame Pratica **- suggerimenti per il 47**

**Se non sapete come fare output sullo standard error,**
**cercate di ridirigere l'output del comando echo sullo standard error**
**prendendo spunto dalla slide intitolata** Ridirezionamenti di Stream di I/O (5)
nel file 4_InterfacciaUtenteACaratteri_BashScripting.pdf