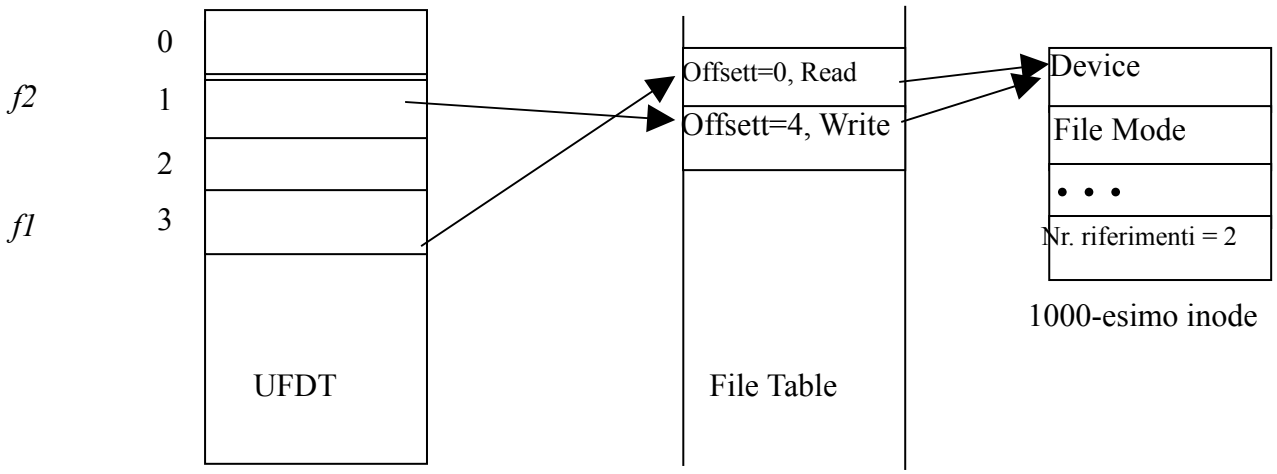
ALTRI ESEMPI DI DOMANDE DELLA SECONDA PROVETTA CON SOLUZIONI

1)Si consideri il seguente frammento di programma:

```
int n;
f1=open("pippo.txt", O_RDONLY);
close(1);
f2=open("pippo.txt", O_WRONLY);
write(f2,&n,sizeof(n));
```

Sapendo che "pippo.txt" corrisponde all' INODE numero 1000, descrivere la situazione della UFDT, FILE TABLE, INODE TABLE, mostrando i valori di offset, numero riferimenti, numero di link, dicendo dove sono localizzati.

Il risultato della esecuzione del codice è illustrato nella seguente figura:



L'effetto del codice e' di scrivere il valore corrente di 'n' nel file 'pippo.txt' anche se la write scrive sulla unità standard Output. I valori di f1 e f2 sono 3 ed 1 rispettivamente. Le strutture dati sono tutte in memoria centrale, area kernel.

2) Si supponga che un file system usi blocchi di 4 Kbyte, e che un puntatore ad un blocco occupi 4 byte. Se l'INODE di un file utilizza 12 blocchi, un blocco indiretto ed un blocco doppiamente indiretto, qual'e' la massima dimensione del file (in byte) ?
(0.8)

Intanto, 12 blocchi rappresentano 12x4kbytes.
I blocco di 4 Kbyte, dato che che un puntatore occupa 4 byte, può contenere 1K puntatori cioe' 1K blocchi, cioè 1Kx4Kbyte.
Un blocco doppio può rappresentare 1Kx1K puntatori, cioè numeri di blocchi dati.
In definitiva: dimensione max file = 12x4K + 1Kx4K + 1Kx1Kx4K = 4KB(12+1K+1Kx1K) = 4.299.210.752 Byte.

3) Rispondere alle seguenti domande:

- Perche' le dimensioni delle pagine sono sempre potenze di due?

Per poter dividere l'indirizzo virtuale in due parole binarie, il numero di pagina e l'0ffsett

4) Si supponga che la tabella delle pagine A[i] di un Sistema Operativo 'SO' con dimensione delle pagine di 4096 byte contenga, in un certo istante, i valori A[i]=2*i per i=0...10, e tutte queste pagine sono valide. Se un processo che gira sotto il controllo di 'SO' produce l'indirizzo virtuale 11223 (in base 10), qual è l'indirizzo fisico (in base 16) corrispondente?

Questo vuol dire che la PMT contiene i seguenti numeri di page frames:

i=0	0
i=1	2
	• • •
i=10	20

*L'indirizzo virtuale 11223 in base 10 è la parola binaria `0010101111010111`. Quindi,visto che la dimensione di pagina è 12 bit (4096 byte), l'indirizzo corrisponde lla pagina virtuale n. 2, che è caricata al page frame 4 (2*i). Quindi l'indirizzo fisico è `0100101111010111` cioè `4BD7` in base 16.*

5) Data la seguente stringa di riferimenti a pagine virtuali: **(1.2)**

1, 7, 5, 1, 2, 1, 7, 1, 5, 7, 2, 1, 5

valutare il Page Fault Rate se il rimpiazzamento è realizzato con gli algoritmi LRU, Ottimo (Belady) e FIFO utilizzando 3 page frame, e una strategia ad Working Set con finestra temporale pari a tre unita' temporali virtuali.

1	7	5	1	2	1	7	1	5	7	2	1	5
1	1	1	1	1	1	1	1	1	1	2	2	2
	7	7	7	2	2	2	2	5	5	5	1	1
		5	5	5	5	7	7	7	7	7	7	5
1	1	1	1	1	1	1	1	5	5	5	5	5
	7	7	7	7	7	7	7	7	7	7	1	1
		5	5	2	2	2	2	2	2	2	2	2
1	1	1	1	2	2	2	2	5	5	5	5	5
	7	7	7	7	1	1	1	1	1	2	2	2
		5	5	5	5	7	7	7	7	7	1	1
1	1	1	1	1	1	1	1	1	1	2	2	2
	7	7	7	2	2	2	-	5	5	5	1	1
		5	5	5	-	7	7	7	7	7	7	5

LRU → PF=9/13=0,69

Belady → PF=6/13=0,46

FIFO → PF=9/13=0,69

WS → PF=9/13=0,69

6) In un Sistema Operativo a memoria contigua ci sono della partizioni libere di 100K, 500K, 200K, 300K, e 600K (nell'ordine). C'è una coda di processi di dimensione 212K, 417K, 112K, e 426K (nell'ordine) che aspettano di essere eseguiti. Utilizzando le strategie First Fit, Worst Fit e Best Fit, qual è la dimensione media delle partizioni lasciate libere? Per quale strategia tutti i

processi possono essere allocati? **(1)**

Con la FirstFit, le partizioni libere sono, nell'ordine: 100K-176K-200K-300K-183K → media= 191.8
La richiesta di 426K non può essere soddisfatta

Con la WorstFit, le partizioni libere sono: 100K-83K-200K-300K-276K → media= 191.8
La richiesta di 426K non può essere soddisfatta

Con la BestFit. Le partizioni libere sono, nell'ordine: 100K-83K-88K-88K-174K → media= 106.l
Tutte le richieste sono soddisfatte.

7) Se la sequenza di indirizzi rilevata da un monitoraggio della esecuzione di un processore è (in base 10): 423-2333-1000-1660-300-2555-1500-425-2330-923-1600-1300, e la dimensione delle pagine è di 512 byte, la dimensione della memoria virtuale è di 65536 byte, rispondere alle seguenti domande:

1. qual'è la sequenza di pagine virtuali?
2. Se la dimensione della PMT è di 6 byte, qual'è la dimensione massima della PMT?

Risposte ai due punti:

1. Con pagine virtuali di 512 byte, gli indirizzi virtuali di inizio-fine delle singole pagine sono:
pg.0=0-511/pg.1=512-1023/pg.2=1024-1535/pg.3=1536-2047/pg.4=2048-2559
Quindi la sequenza di riferimenti virtuali e': 0, 4, 1, 3, 0, 4, 2, 0, 4, 1, 3, 2

*2. Visto che la memoria virtuale è di 65536 byte, tale e' lo spazio d'indirizzamento. Quindi ci sono 65536/512 = 128 pagine virtuali al massimo per ogni processo. Dato che ogni elemento della PMT e' di 6 byte, la dimensione massima è 128 * 6=768 byte*

8) Per la stringa dei riferimenti alle pagine virtuali del precedente punto 7) :

0	4	1	3	0	4	2	0	4	1	3	2

Se ci sono tre page frame nella memoria fisica, i tempi medi di accesso per byte nella memoria primaria (fisica) e nella memoria secondaria (disco) sono rispettivamente Tm=600ns e Td=44.5ms per byte, dare il tempo medio di accesso per byte della memoria virtuale per i rimpiazzamenti ottimo (o di Belady), FIFO e LRU.

Per rispondere a questa domanda bisogna vedere qual'e' la percentuale di Page Fault con la stringa in esame. Innanzitutto si osservi che ogni processo ha a disposizione 1536/512 = 3 pagine fisiche (page frames).

Partiamo con Belady, rappresentando il contenuto effettivo della memoria fisica.

Stringa d'ingresso	*0*	*4*	*1*	*3*	*0*	*4*	*2*	*0*	*4*	*1*	*3*	*2*
Page Frame 0	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*1*	*1*	*1*
Page Frame 1		*4*	*4*	*4*	*4*	*4*	*4*	*4*	*4*	*4*	*3*	*3*
Page Frame 2			*1*	*3*	*3*	*3*	*2*	*2*	*2*	*2*	*2*	*2*
Page fault	*Si*	*Si*	*Si*	*Si*	*No*	*No*	*Si*	*No*	*No*	*Si*	*Si*	*No*

Probabilita' di page fault: PF=7/12=0.5833

Vediamo ora FIFO:

Stringa d'ingresso	*0*	*4*	*1*	*3*	*0*	*4*	*2*	*0*	*4*	*1*	*3*	*2*
Page Frame 0	*0*	*0*	*0*	*3*	*3*	*3*	*2*	*2*	*2*	*2*	*2*	*2*
Page Frame 1		*4*	*4*	*4*	*0*	*0*	*0*	*0*	*0*	*1*	*1*	*1*
Page Frame 2			*1*	*1*	*1*	*4*	*4*	*4*	*4*	*4*	*3*	*3*
Page fault	*Si*	*Si*	*Si*	*Si*	*Si*	*Si*	*Si*	*No*	*No*	*Si*	*Si*	*No*

Probabilità di page fault: PF=9/12=0.75

Vediamo infine LRU:

Stringa d'ingresso	*0*	*4*	*1*	*3*	*0*	*4*	*2*	*0*	*4*	*1*	*3*	*2*
Page Frame 0	*0*	*0*	*0*	*3*	*3*	*3*	*2*	*2*	*2*	*1*	*1*	*1*
Page Frame 1		*4*	*4*	*4*	*0*	*0*	*0*	*0*	*0*	*0*	*3*	*3*
Page Frame 2			*1*	*1*	*1*	*4*	*4*	*4*	*4*	*4*	*4*	*2*
Page fault	*Si*	*Si*	*Si*	*Si*	*Si*	*Si*	*Si*	*No*	*No*	*Si*	*Si*	*Si*

Probabilità di page fault: PF=10/12=0.833

Visto che il tempo medio per byte e' dato dalla: T=PF(Tm+Td)+(1-PF)Tm (dove Tm e Td sono il tempo medio in memoria fisica e di disco, cioe' Tm=600ns e Td=44.5ms per byte), si ha che:

Belady → T=25.96 ms
FIFO → T=33.37 ms
LRU → T=37.07 ms

Quindi, le strategie Belady, FIFO portano ad un tempo medio inferiore a 34 ms.
Anche se non è richiesto, è stato visualizzato per completezza il contenuto della memoria fisica per tutti i valori della stringa d'ingresso.

Inoltre, per completezza, vengono visualizzate le PMT per la stringa di pagine virtuali 0, 4, 1, 3, 0, 4, 2. Visto che si arriva al massimo alla pagina virtuale 4, e' sufficiente mostrare le righe 0-4:

Sequenza delle PMT al variare dei riferimenti alle pagine virtuali della stringa di ingresso:

rif. virtuale 0 | *rif. virtuale 4* | *rif. virtuale 1*

pg. Virtuale	*V*	*Page frame*	*Indirizzo Disco*	*pg. virtuale*	*V*	*Page frame*	*Indirizzo Disco*	*pg. virtuale*	*V*	*Page frame*	*Indirizzo Disco*
0	*1*	*0*	*--*	*0*	*1*	*0*	*--*	*0*	*1*	*0*	*--*
1	*0*		*--*	*1*	*0*		*--*	*1*	*1*	*2*	*--*
2	*0*		*--*	*2*	*0*		*--*	*2*	*0*		*--*
3	*0*		*--*	*3*	*0*		*--*	*3*	*0*		*--*
4	*0*		*--*	*4*	*1*	*1*	*--*	*4*	*1*	*1*	*--*

rif. virtuale 3

pg. Virtuale	*V*	*Page frame*	*Indirizzo Disco*
0	*0*	*0*	*--*
1	*1*	*2*	*--*
2	*0*		*--*
3	*1*	*0*	*--*
4	*1*	*1*	*--*

rif. virtuale 0

pg. virtuale	*V*	*Page frame*	*Indirizzo Disco*
0	*1*	*1*	*--*
1	*1*	*2*	*--*
2	*0*		*--*
3	*1*	*0*	*--*
4	*0*	*1*	*--*

rif. virtuale 4

pg. Virtuale	*V*	*Page frame*	*Indirizzo Disco*
0	*1*	*1*	*--*
1	*0*	*2*	*--*
2	*0*		*--*
3	*1*	*0*	*--*
4	*1*	*2*	*--*

rif. virtuale 2

pg. virtuale	*V*	*Page frame*	*Indirizzo Disco*
0	*1*	*1*	*--*
1	*0*	*2*	*--*
2	*1*	*0*	*--*
3	*0*	*0*	*--*
4	*1*	*2*	*--*

Quest'ultima PMT descrive i primi due campi (cioe' bit V + page frame) dopo il 7° riferimento.

9) Se la dimensione della pagina virtuale è di 512 byte, e l'ultimo riferimento virtuale 4 della stringa corrisponde all'indirizzo 2333, qual'è il corrispondente indirizzo fisico?

L'indirizzo 2322 corrisponde alla pagina virtuale 4 e all'offset in pagina 274. La pagina virtuale 4 e' caricata nel page frame 2.
*Quindi l'indirizzo fisico e': |_____ 2|_ 274| cioe' 2*512 + 274 = 1298*

10) Ci sono 4 processi, tutti arrivati all'istante 0, nell'ordina: P1 con durata 12 unità temporali, P2 che richiede 2 unità, P3 di 7 unità, P4 di 1 unità temporale. Calcolare il tempo di turnaround medio con le strategie di schedulazione FCFS (first come first served), SJF (shortest job first), RR (round robin)

Con la schedulazione FCFS il tempo di turnaround medio e': (12+14+21+22)/4= 17.25.

Con la schedulazione SJF il tempo di turnaround medio e': (1+3+10+22)/4=9 corrispondenti alla sequenza di schedulazione P4-P2-P3-P1

Con la schedulazione Round Robin con quanto di 1, la schedulazione risulta:

```
P1-P2-P3-P4-P1-P2-P3-P1-P3-P1-P3-P1-P3-P1-P3-P1-P3-P1-P1-P1-P1-P1
          ^     ^                                 ^                 ^
```

dove ^ indica il punto di terminazione.
Quindi: P4 termina all'istante 4. P2 termina all'istante 6, P3 termina all'istante 17 e P1 termina all'istante 22.
Quindi il tempo di turnaround medio e' (4+6+17+22)/4=12.25

11) Descrivere gli identificatori numerici associati ad un file e descrivere gli identificatori numerici associati ad un processo

identificativi del file:

i. *numero di Inode*

ii. *UID e GID del proprietario del file*

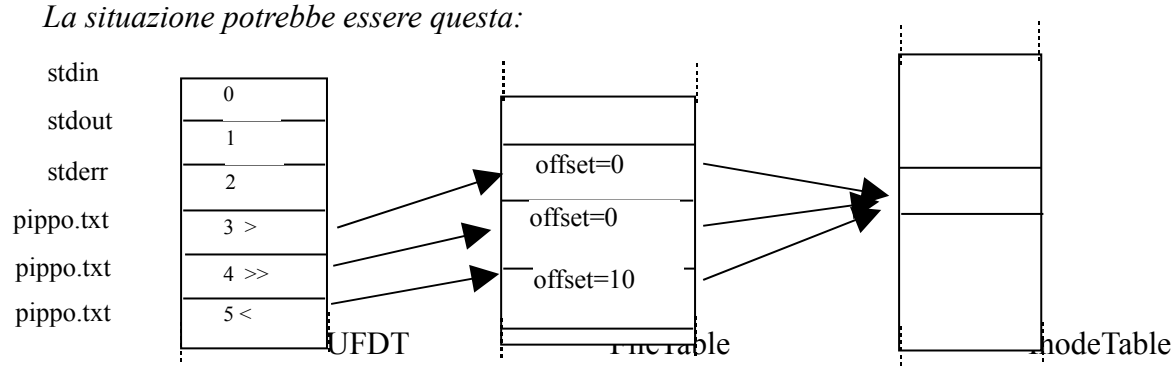
identificatidel processo:

iii. *Process ID*
iv. *Parent Process ID*
v. *Real e Effective Process User ID*
vi. *Real e Effective Process Group ID*

Questi ultimi sono usati per le protezioni ai file

12) Si descrivano le tabelle utilizzate dal file system della famiglia Unix (user file descriptor table, file table, inode table) nel caso che un programma apra il file 'pippo.txt' tre volte in modalita' diverse (rispettivamente scrittura, accodamento e lettura) e dopo che all'ultima apertura del file segua una lettura di 10 byte.

La situazione potrebbe essere questa:

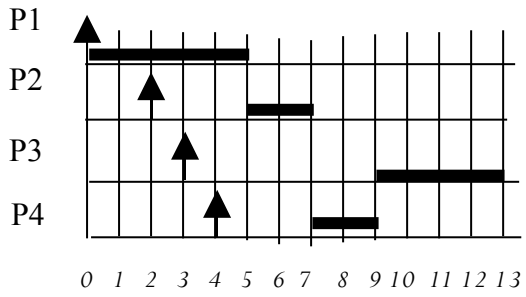


13) Si supponga che in un Sistema Operativo vengano creati tre processi, ciascuno negli istanti d'arrivo e con le durate indicate:

Processo	Istante d'arrivo	Durata
P1	0	5
P2	2	2
P3	3	4
P4	4	2

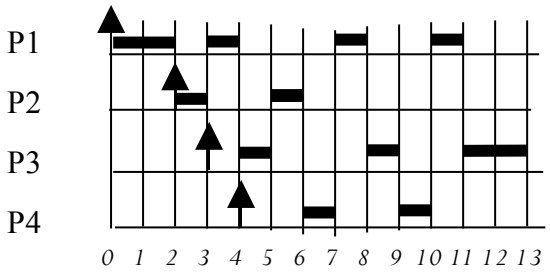
Si calcoli il tempo di turnaround medio con le strategia di schedulazione Shortest Job First (o Shortest-Process-Next (SPN)) e Round-Robin (RR per un quanto pari a 1.

Con la schedulazione SPN si ha:



e quindi: turnaround_medio=[(5-0)+(7-2)+(13-3)+(9-4)]/4=6.25

mentre con la schedulazione RR si ha:



Quindi,
P1 finisce in 11
P2 finisce in 6
P3 finisce in 13
P4 finisce in 10

quindi: turnaround_medio=[(11-0)+(6-2)+(13-3)+(10-4)]/4=7.75

14) Si abbiano i file `pippo.txt` e `pluto.txt` con le protezioni `600` e `400` rispettivamente. Si eseguono poi i comandi

```
ln –s pippo.txt pippo1.txt ; chmod 304 pippo.txt ; chmod 774 pippo1.txt.
ln pluto.txt pluto1.txt ; chmod 304 pluto.txt ; chmod 774 pluto1.txt
```

Può il proprietario di `pippo.txt` eseguire il comando `cat pippo1.txt`? può un altro utente non appartenente allo stesso gruppo eseguire lo stesso comando ?
Può il proprietario di `pluto.txt` eseguire il comando `cat pluto1.txt`? può un altro utente non appartenente allo stesso gruppo eseguire lo stesso comando ?

Inizialmente il file `pippo.txt` ha le protezioni che consentono al proprietario del file di leggere e scrivere mentre il proprietario di `pluto.txt` può solo leggerlo. Né altri utenti dello stesso gruppo né altri possono fare altro. In seguito (dopo i comandi chmod) i due file possono essere scritti ed eseguiti dal proprietario e letti da tutti gli altri utenti, mentre gli utenti appartenenti al gruppo non possono fare niente (304). Il file `pippo1.txt` è un link simbolico a `pippo.txt`; il link può essere letto dal proprietario ma quando egli cerca di leggere il file puntato vede che il file non ha i permessi di lettura.
Quindi il proprietario di `pippo.txt` non può eseguire `cat pippo1.txt`, mentre un utente non appartenente allo stesso gruppo può farlo.
Per quanto riguarda `pluto1.txt`, essendo un hard link, ogni operazione sul file originale `pluto.txt` si traduce direttamente sul file linkato e viceversa. In particolare, il chmod sul file linkato passa direttamente sul file originale.
Quindi il proprietario di `pluto.txt` può eseguire `cat pluto1.txt`, così come un altro utente non appartenente al gruppo.
*In definitiva le risposte sono **NO-SI/SI-SI***