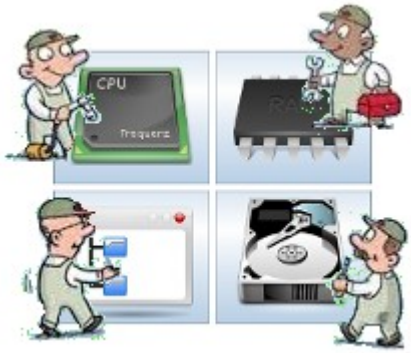
Laboratorio di Sistemi Operativi

esperienze con simulatori

2014.11



Indice

Introduzione e generalità su OS Sim

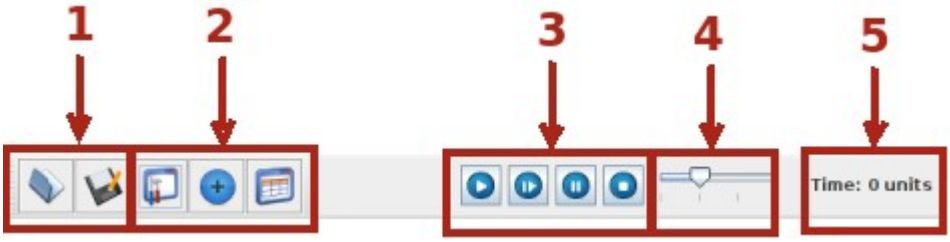
Il software OS Sim (Operating System Concepts Simulator) è una applicazione che simula graficamente i concetti che stanno alla base di un Sistema Operativo allo scopo di essere di aiuto nella loro comprensione.

OS Sim è sviluppato in Java e disponibile quindi in qualsiasi sistema in cui sia installata una JVM e può essere scaricato da `http://sourceforge.net/projects/oscsimulator/`. Il software è sviluppato all'interno dell'associazione FIB Alumni che raggruppa alunni, che hanno frequentato la facoltà di Informatica, e insegnanti della Universitat Politècnica De Catalunya.

I moduli simulati su cui si possono fare esperienze (accessibili per mezzo di quattro grossi pulsanti presenti nella schermata *Wellcome* o selezionandoli ognuno dall'apposita voce di menù) riguardano: la gestione dei processi, la memoria centrale (la cui simulazione di gestione verrà tuttavia trattata in questi appunti utilizzando un altro simulatore di cui si parlerà nel momento opportuno), il filesystem, l'accesso ai dati su disco. Il software è corredato, per ogni modulo, da numerosi esempi ed esercizi pronti, modificabili all'occorrenza, per essere osservati e, inoltre, è possibile impostare anche simulazioni completamente personalizzate. È presente anche un sistema di aiuto relativo a ogni fase del lavoro. Durante la simulazione, in ambedue i software, è possibile intervenire manualmente cambiando preferenze o aggiungendo nuovi elementi e osservandone il comportamento.

Per una completa comprensione delle simulazioni descritte in questi appunti è necessario avere conoscenze su concetti trattati nella dispensa ***atapSO*** i cui capitoli, relativi alle singole parti su cui si propone la simulazione, sono da considerarsi propedeutici alle simulazioni stesse. Questa dispensa può essere considerata inoltre come complemento di atapSO trattando anche concetti non presenti in quella.

Qualunque sia la simulazione scelta in OS Sim, viene visualizzata una barra di pulsanti di scelta rapida con identica funzione.



I pulsanti del gruppo **1**, come comune, possono essere utilizzati per caricare da disco una simulazione conservata in precedenza o salvare su disco quella attuale.

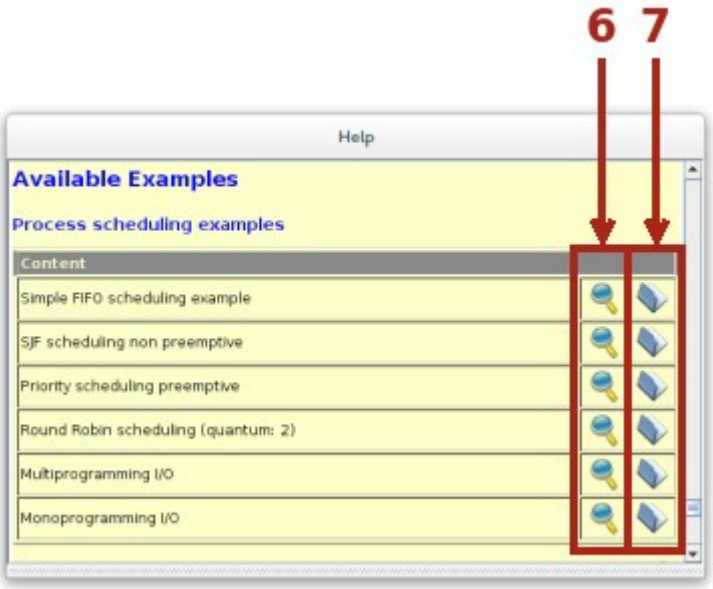
Nel gruppo **2** con il primo pulsante da sinistra (*Settings*) si modificano le caratteristiche della simulazione: si cambia la politica di gestione dei processi, della memoria, del modulo di cui si sta eseguendo la simulazione. Con il pulsante centrale (*Add ...*) si può aggiungere un elemento all'ambiente simulato: un ulteriore processo, un'altra operazione ... Il pulsante sulla destra mostra una finestra con i risultati statistici della simulazione effettuata (*Data and statistics*).

Il gruppo di pulsanti **3** regola l'esecuzione della simulazione. Il primo pulsante da sinistra avvia la simulazione che si può mettere in pausa con il secondo pulsante da destra o fermare (ultimo pulsante a destra). Il secondo pulsante da sinistra avvia un singolo passaggio della simulazione e si possono osservare i cambiamenti conseguenti.

Il cursore **4** regola la velocità di esecuzione della simulazione quando si avvia con il pulsante a sinistra di **3**.

Nella parte destra della barra dei pulsanti (**5**) è visualizzato il tempo cui si è arrivati nella simulazione misurato in unità di lavoro della CPU.

Tutte le esperienze descritte partono da esempi già presenti nel software a cui si aggiungono, in un caso, elementi significativi per le valutazioni successive. Gli esempi sono accessibili selezionandoli dal menù *Help* e quindi *Examples* e subito dopo il tipo di simulazione scelta.

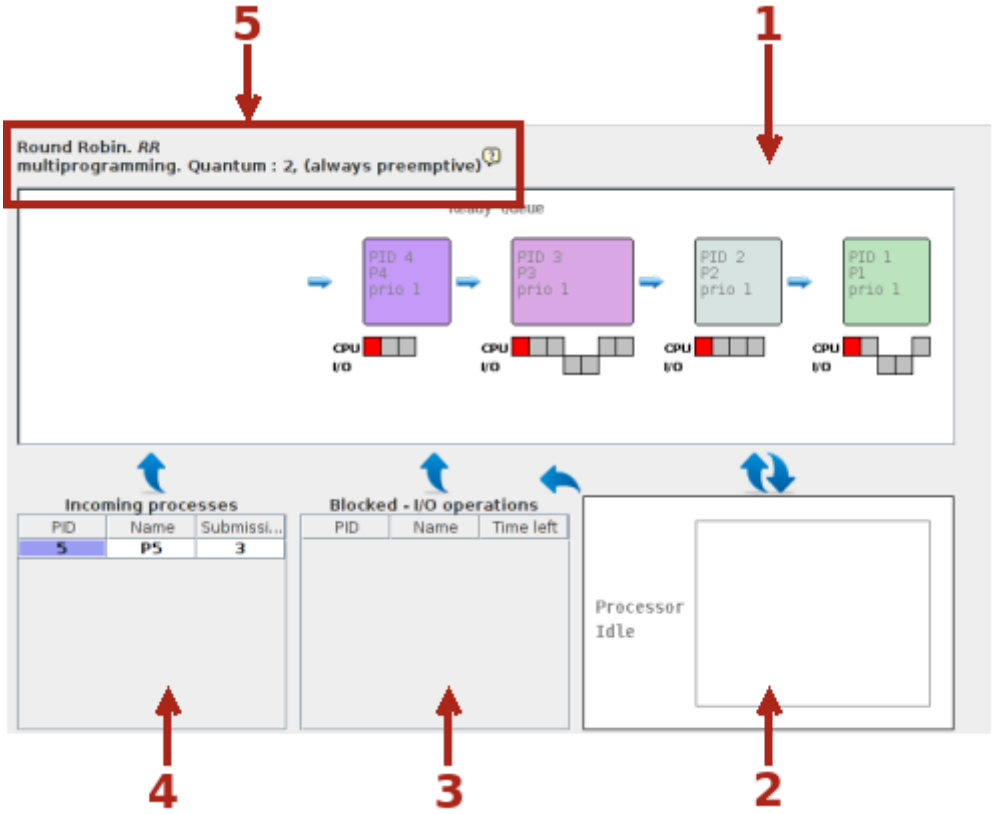


Ogni esempio proposto, a prescindere dalla simulazione scelta, presenta sulla destra due pulsanti: quello appartenente al gruppo **6** visualizza la finestra di help, una sotto-finestra con la spiegazione dell'esempio proposto e la specificazione degli oggetti definiti nell'esempio.

Il pulsante appartenente al gruppo **7** carica l'esempio nell'emulatore pronto per essere personalizzato o eseguito così come è stato impostato.

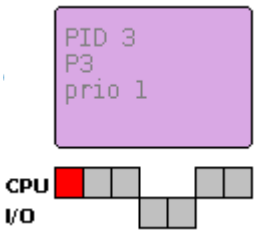
Scheduling dei processi

Per studiare le proprietà delle politiche di scheduling dei processi si può partire da *Help*, *Examples*, *Process scheduling* e scegliere fra gli esempi proposti *Multiprogramming I/O*. Si tratta della simulazione dei concetti su cui si basa un S.O. multiprogrammato con processi che richiedono operazioni di I/O. Sulla base di questo esempio si apporteranno delle modifiche.



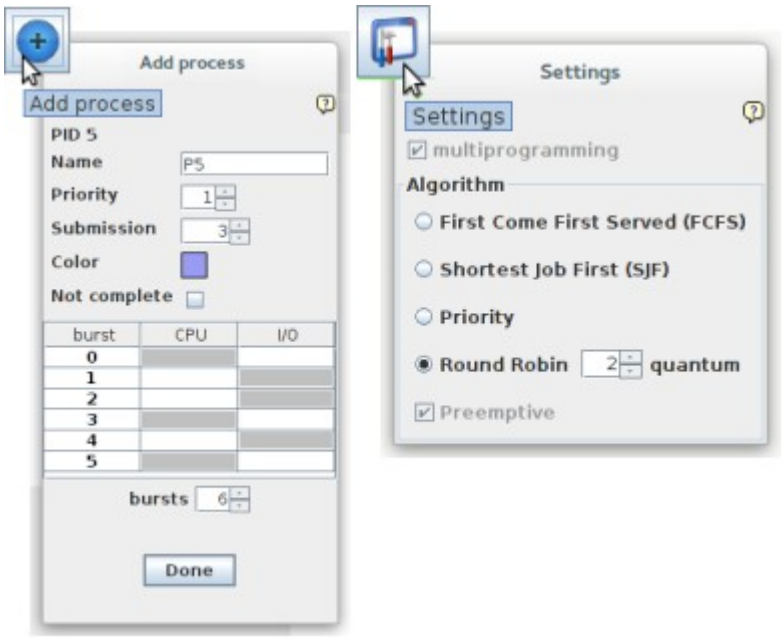
Le parti in cui è suddivisa la finestra rispecchiano gli stati di un processo per come sono esemplificati nel programma di simulazione. La coda dei *Ready* (**1**), il processo in *Running* (**2**) con evidenziate le transizioni nei due sensi (da **1** a **2** e viceversa). Lo stato *Blocked* (**3**) in cui transitano i processi in attesa del completamento delle operazioni I/O, completate le quali si transita nella coda *Ready*, e la coda degli *Incoming* (**4**) dove prendono posto i processi in attesa del loro turno di transizione allo stato di *Ready* quando il tempo trascorso dall'avvio coincide con quello di presentazione.

Nella parte alta a sinistra (**5**) è riportata la politica di scheduling che verrà applicata. Accanto alla descrizione c'è una piccola icona con un punto interrogativo che, cliccato, abilita la visualizzazione di una finestra di help con la descrizione della politica adottata.



Ogni processo è rappresentato da un rettangolo colorato che contiene i propri dati identificativi. Nella parte sottostante è specificata la durata in termini di unità base (*bursts*) rappresentate con quadratini grigi distinti in base alle competenze delle risorse che se ne devono occupare: il processo con PID 3 necessita, per la sua esecuzione completa, di tre unità di CPU, due di I/O e infine altre due di CPU. Il quadratino rosso indica il punto in cui è arrivato il processo nella sua evoluzione.

Rispetto all'esempio proposto dal software sono state apportate due modifiche:

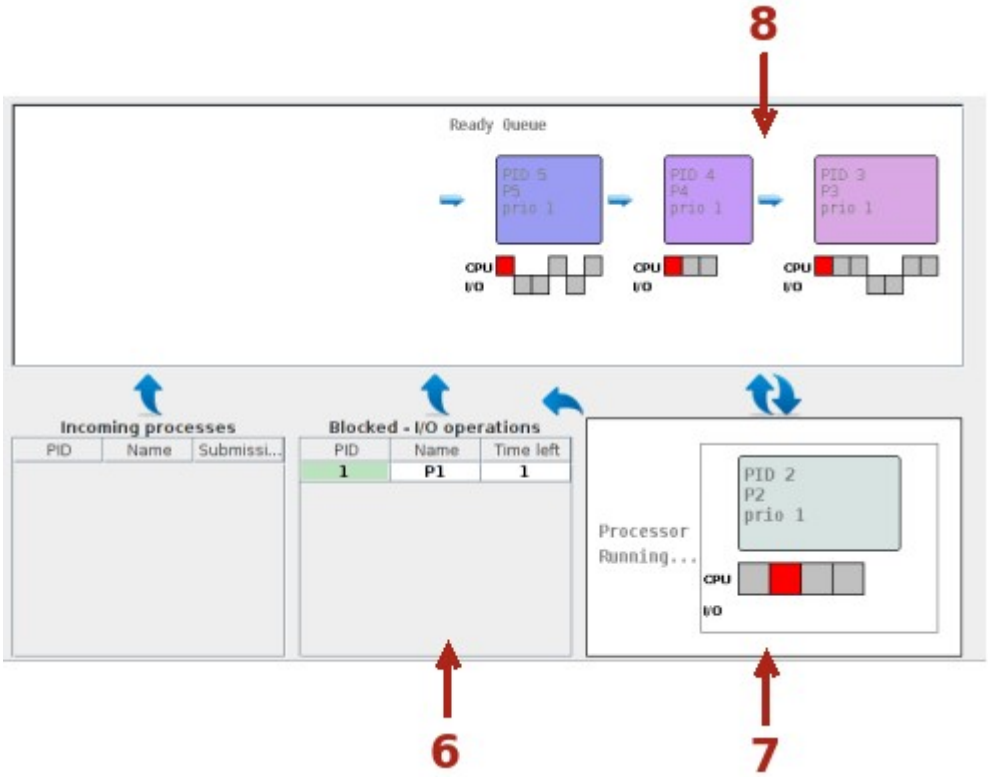


1. è stato aggiunto un processo. Le caratteristiche da specificare sono: *Name* (nome mnemonico), *Priority* (priorità: ad un numero più grande corrisponde una priorità maggiore), *Submission* (tempo, dall'avvio, in cui deve essere avviato il processo), *Color* (colore, casuale ma si può sceglierne uno diverso selezionando il pulsante, che fa riconoscere visivamente il processo), *Not complete* (indica se il processo non arriva a terminazione. Può essere per esempio un processo di sistema che deve rimanere sempre attivo). I *bursts* indicano le unità di tempo necessarie per completare il processo. In dipendenza della quantità scelta vengono inserite l'equivalente numero di righe nella tabella. Cliccando sulla casella si può scegliere se si tratta di un frammento di utilizzo di CPU o I/O
2. è stata modificata la politica di scheduling. È stata scelta la Round Robin con un *time slice* di due unità di tempo (*quantum*). Si tratta, come evidenziato nella finestra, di una politica di tipo *preemptive* (a prelazione): è il S.O. che si occupa di togliere la risorsa CPU al processo dopo che è trascorso il time slice assegnato o quando c'è una richiesta di operazioni I/O.

A questo punto si può avviare la simulazione. Conviene per una più attenta osservazione un procedimento di tipo step-by-step in modo da osservare in base alla presenza dei processi in *Ready*

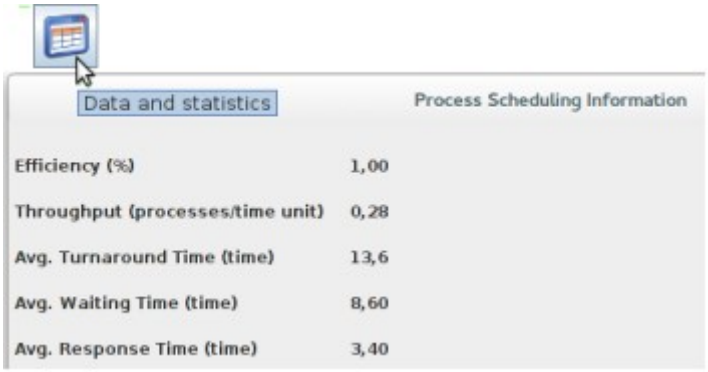
e alle loro caratteristiche, come si evolve il sistema.

Per esempio se si osserva il sistema quando sono trascorse 3 unità di tempo da un raffronto con lo screen shot precedente, si ha la seguente situazione:



Il processo con PID 1 ha richiesto operazioni I/O (il terzo burst), il S.O. ha tolto la risorsa CPU e lo ha messo in stato *Blocked* (**6**) da cui uscirà, per tornare alla coda *Ready* quando le operazioni termineranno (dopo 2 bursts). Nel frattempo il processo con PID 2 è transitato in *Running* (**7**) e ha già eseguito un burst (il time slice è di due bursts), il processo con PID 5 è stato inserito nella coda *Ready* (**8**).

Terminata la simulazione si possono avere informazioni di tipo statistico sulla sessione di lavoro. La finestra delle informazioni si può considerare suddivisa in due parti:



la parte superiore riporta parametri riguardanti la simulazione nel suo complesso: *Efficiency* (percentuale in cui la CPU è stata impegnata), *Throughput* (quantità di processi completati per unità di tempo), *Avg. Turnaround* (tempo medio di completamento dei processi), *Avg. Waiting* (tempo

medio di attesa dei processi nella coda Ready), *Avg. Response* (tempo medio di risposta del sistema: quanto tempo passa da quando i processi vengono sottomessi al sistema a quando *vengono presi in considerazione* dalla CPU).

La parte sottostante della finestra mostra informazioni riferite al singolo processo esaminato:

PID	Name	Priority	Submission
1	P1	1	0
2	P2	1	0
4	P4	1	0
3	P3	1	0
5	P5	1	3

...

Response	Waiting	Turnaround	% CPU	% IO
0	5	10	0.375	0.4
2	8	12	0.33333...	0.0
6	11	14	0.21428...	0.0
4	10	17	0.33333...	0.28571...
5	9	15	0.25	0.5

la colonna PID mostra la sequenza di terminazione dei processi. Le informazioni *Response*, *Waiting* e *Turnaround* questa volta sono riferiti al singolo processo. Le ultime due colonne a destra indicano le caratteristiche del singolo processo in termini di percentuale di bursts di occupazione CPU e di occupazione di I/O.

Anche se la simulazione non prevede correzioni alla politica di Round Robin (per esempio gestione priorità dinamica), si possono apprezzare le differenze di politica esaminando e confrontando i risultati ottenuti quando si sceglie una politica di scheduling diversa. Per esempio se si lascia la stessa configurazione dei processi, si può scegliere come politica di scheduling FCFS (First Come First Served) di tipo non preemptive.

Efficiency (%)	0,95
Throughput (processes/time unit)	0,26
Avg. Turnaround Time (time)	12,8
Avg. Waiting Time (time)	7,80
Avg. Response Time (time)	5,20

PID	Name
2	P2
4	P4
1	P1
3	P3
5	P5

...

Response	Waiting	Turnaround
2	2	6
9	9	12
0	9	14
6	9	16
9	10	16

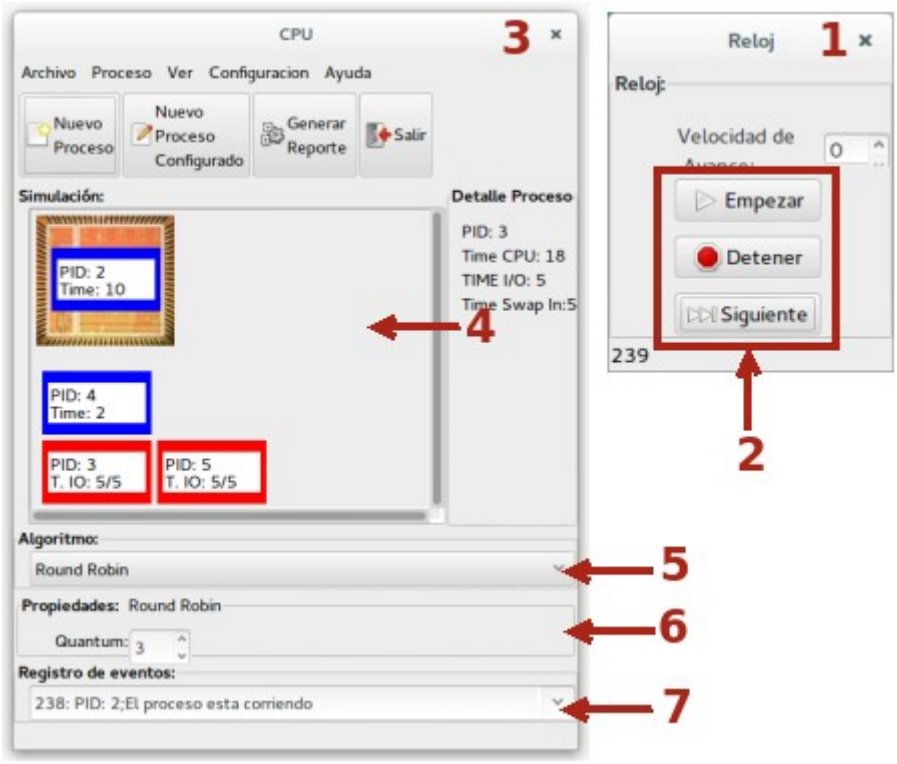
Da un esame comparativo si può arrivare alle conclusioni:

1. il RR ha tempi di risposta migliori. Questo tipo di politica è stata studiata per questo scopo: per i sistemi di tipo real-time.
2. Anche se il turnaround, in media, è migliore per FCFS, si consideri che qui RR è applicato in forma *pura*, senza aggiustamenti. I processi 2 e 4 non impiegano risorse I/O e vengono eseguiti più rapidamente. Affinché si notino di più le differenze basterebbe effettuare una simulazione con 2 processi: uno che non impiega risorse I/O e uno che ne fa un uso intensivo ed eseguire due simulazioni scegliendo le politiche RR e FCFS
3. l'efficienza è maggiore per RR

Generalità su OVSOS

OVSOS (Other Visual Simulation of an Operating System) è un software scritto in Python, e scaricabile da `http://ovsos.sourceforge.net/`, che simula in un ambiente visuale il funzionamento di un sistema operativo che gestisce la multiprogrammazione e la paginazione con memoria virtuale.

All'avvio il software presenta tre finestre:



La finestra *Reloj* (orologio **1**) contiene il gruppo di pulsanti **2** che regolano l'avvio della simulazione, la cui velocità è gestita per mezzo del cursore sovrastante. Il pulsante più importante, al solito, è quello più in basso che permette l'esecuzione di un singolo step. Nella parte inferiore sinistra della finestra è indicato l'unità di tempo cui si è arrivati.

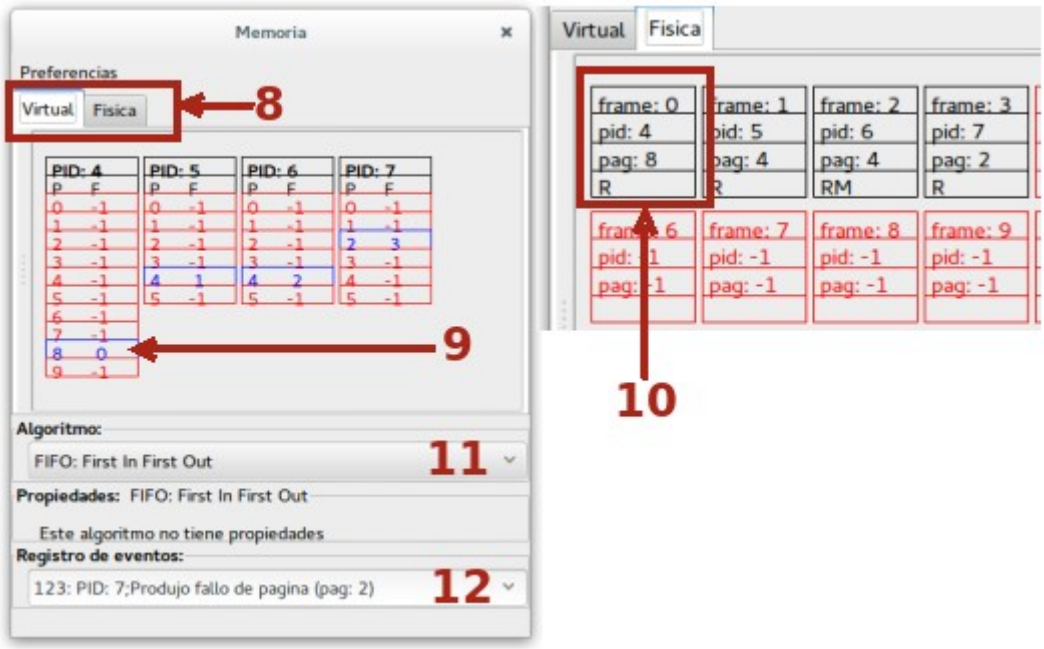
La finestra *CPU* (**3**), a parte la presenza dei pulsanti allocati nella parte superiore di cui si esamineranno le funzioni più avanti, presenta una parte centrale (**4**) con visualizzato lo stato di avanzamento dei processi: il processo in stato di *Running* viene rappresentato sopra la CPU e delimitato da un riquadro di colore blu (nella figura quello identificato dal PID 2). Oltre all'identificativo nel riquadro è indicato il tempo (in unità di lavoro CPU) necessario per portare a terminazione il processo. Subito sotto la CPU, e sempre contornati da un riquadro blu, sono mostrati i processi nella coda di *Ready*. Il processo con PID 3 della figura è bloccato (è contornato da un riquadro rosso) in attesa del completamento di una operazione I/O. In questo caso è visualizzato il tempo necessario affinché il processo termini l'operazione I/O che, nel simulatore, riguarda la richiesta di una nuova pagina da caricare in memoria centrale. Quando si seleziona un processo, sulla destra vengono mostrate informazioni su di esso.

Il menù a discesa **5** permette di scegliere la politica di scheduling e se la politica scelta ha bisogno di ulteriori specificazioni, queste vanno specificate in **6**. Per esempio se si sceglie RR è necessario

specificare i quantum del time slice.

È presente anche un registro degli eventi (**7**) per i messaggi. Nella figura il messaggio indica che nell'istante 238 il processo con PID 2 è in Running.

La terza finestra mostra una visione della gestione della memoria.



La finestra presenta due schede (**8**) una per la memoria virtuale e una per quella fisica. Nella scheda della memoria virtuale sono mostrate le tabelle di mappatura delle pagine dei processi: le pagine presenti in memoria centrale sono evidenziate come righe blu, quelle in memoria virtuale in rosso. Se, per esempio, si prende in esame il processo con PID 4 si può notare che la pagina 8 è allocata nel frame 0 (**9**). Nella pagina della memoria fisica si riscontra (**10**) che il frame 0 è assegnato alla pagina 8 del processo con PID 4. L'indicatore R segnala che la pagina è utilizzata in lettura e, nella necessità di uno swap-out, non è necessario riscriverla, cosa che invece si rende necessario se la pagina è stata modificata (RM). Nella simulazione la memoria fisica è suddivisa in 24 frame. Le pagine che non sono allocate in memoria hanno nella colonna del frame (*F*) della memoria virtuale il valore -1, così come, nella memoria fisica, i frame nei quali non sono allocate pagine di un qualche processo (nel frame sia *pid* che *pag* contengono il valore -1).

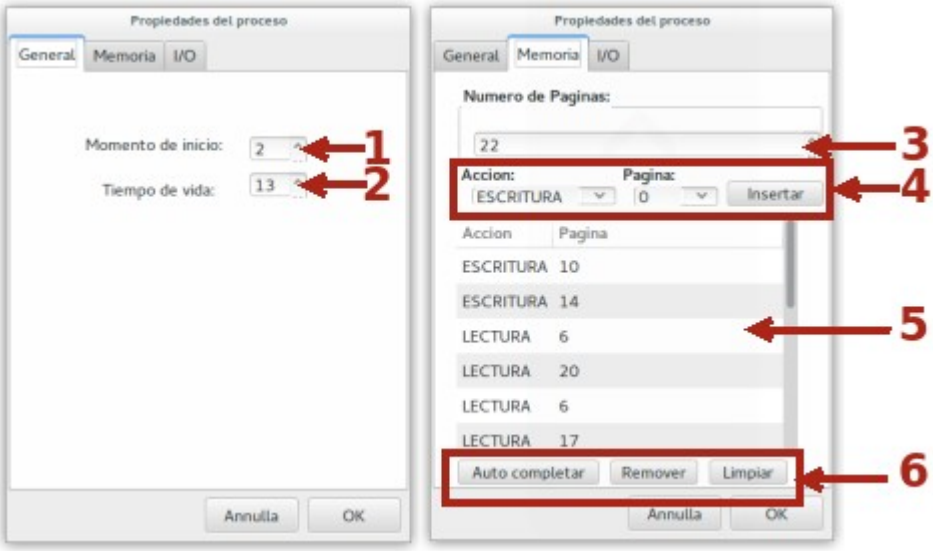
In **11** è specificato l'algoritmo utilizzato per lo swapping delle pagine (attualmente è previsto solo FIFO). Anche in questa finestra è previsto (**12**) il registro degli eventi: istante dell'evento, PID del processo, pagina richiesta.

Swapping delle pagine

La gestione dello swapping è osservabile più chiaramente se si scelgono i processi in modo che la quantità di pagine totali di cui sono composti superi il numero di frame disponibili nel simulatore (24). Finché ci sono frame disponibili e il processo fa richiesta di una pagina, si genera un evento di tipo *Page fault* e la pagina viene caricata nel primo frame disponibile. Se invece la memoria è tutta occupata la gestione del page fault è più complessa: secondo la politica adottata (nel simulatore solo FIFO) si sceglie la pagina più vecchia che, quindi, è la candidata ad essere sostituita e lo può essere

subito se la pagina era a sola lettura, a meno che non sia una pagina modificata e, in tal caso, bisogna prima registrare la pagina modificata e solo dopo può essere sostituita.

Per eseguire una simulazione è necessario definire i processi. I pulsanti *Nuevo Proceso* e *Nuevo Proceso Configurado* disponibili nella finestra CPU consentono tale operazione. Il primo pulsante permette la generazione di un processo in cui tutte le proprietà devono essere specificate. Può essere comodo tuttavia cominciare da processi già configurati anche se, in ogni caso, tutte le proprietà possono essere modificate (*Nuevo Proceso Configurado*).



Le proprietà del processo sono racchiuse in due schede (la terza, prevista per ulteriori versioni, non è attiva) della finestra visualizzata in seguito alla pressione di uno qualsiasi dei pulsanti per la generazione di un processo.

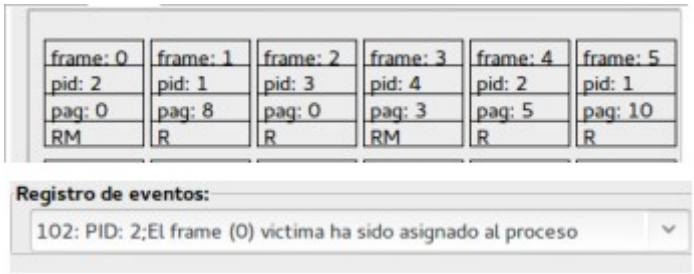
La scheda *General* comprende la definizione del momento di inizio del processo (**1**) e del tempo di vita (**2**) espresso in unità di lavoro della *CPU*. Se il processo è già configurato i due campi hanno già valori.

Nella scheda *Memoria* si può stabilire la quantità di pagine (**3**) e, per ogni unità di tempo di vita del processo, la pagina da utilizzare e le modalità di utilizzo della pagina (**4**). Il pulsante *Insertar* a destra di **4** inserisce la scelta effettuata nell'elenco degli accessi alle pagine (**5**). Si può lasciare al programma, anche se non si è scelta una configurazione già pronta, di riempire l'elenco **5** selezionando il primo pulsante a sinistra di **6** (*Auto completar*). il pulsante centrale di **6** elimina la riga selezionata dall'elenco **5**, il pulsante a destra pulisce l'intero elenco.

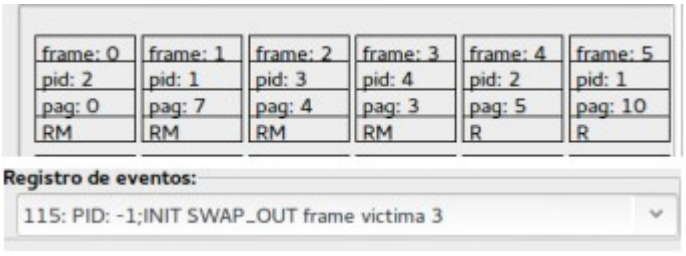
Definendo alcuni processi in modo da avere un numero di pagine in totale maggiore del numero di frame della memoria fisica si potrebbe avere la situazione descritta in seguito. Naturalmente quando si scelgono processi già configurati le proprietà sono settate in modo casuale ma anche se la situazione dell'esercitazione effettuata è diversa da quella descritta negli screen shot, rimangono valide le considerazioni qui espresse anche se i tempi saranno sicuramente diversi.

Finché ci sono disponibili frame le pagine richieste dai processi le occuperanno.

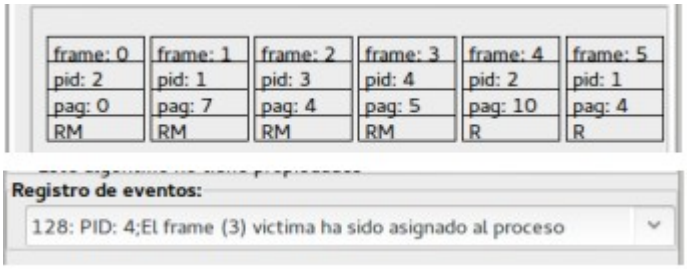
Al tempo 95 c'è stato un page fault e l'ultimo frame disponibile (23) è stato assegnato alla pagina 6 del processo con PID 4. Da questo momento per ogni richiesta di nuova pagina si dovrà liberare un frame sostituendo la pagina che attualmente lo occupa. Per la politica adottata dal simulatore (FIFO) i frame da assegnare cominceranno dal frame 0 e, a seguire, i successivi.



Al tempo 102 il frame 0 è assegnato alla pagina 0 del processo con PID 2. La pagina che era nel frame era a lettura (screen shot precedente) e lo swapping si è potuto effettuare immediatamente in seguito alla richiesta. Anche per l'assegnazione dei frame 1 e 2 non ci sono problemi essendo assegnati in precedenza a pagine non modificate.

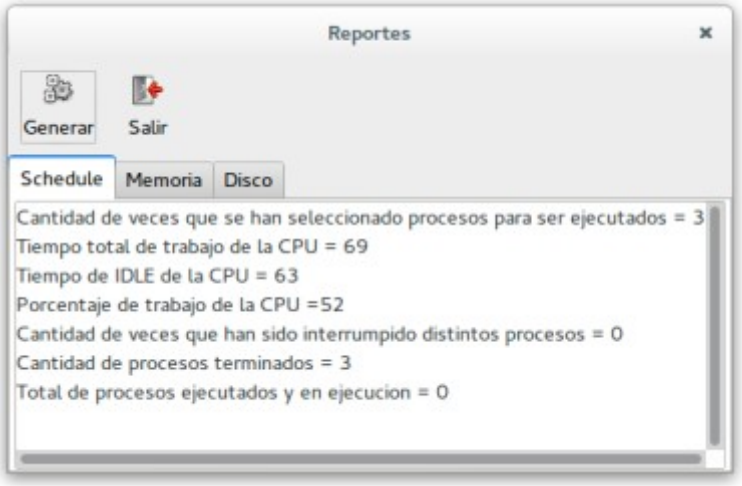


Al tempo 115 è richiesto il caricamento di una nuova pagina e, in accordo con la politica di allocazione, il frame destinazione dovrebbe essere il 3 ma questo è assegnato ad una pagina modificata e quindi, come riporta il registro degli eventi, è necessario uno swap_out.



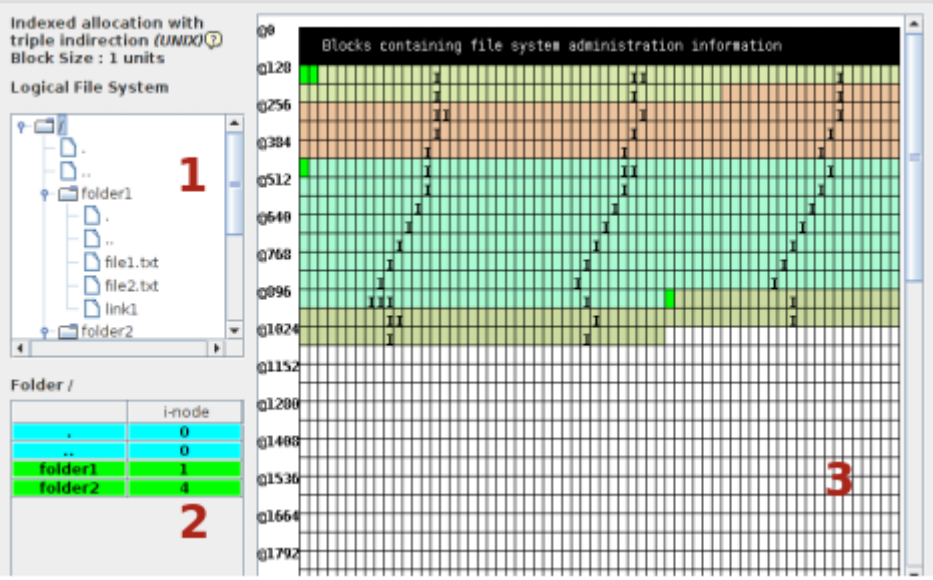
Infine al tempo 128, conclusa l'operazione di salvataggio della pagina modificata, il frame 3 può essere assegnato al processo PID 4. Nel frattempo, come si può notare da un raffronto con lo screen shot precedente, sono stati assegnati i frame 4 e 5 che ospitavano pagine non modificate.

Il pulsante *Generar Reporte* della finestra *CPU* visualizza una finestra con informazioni varie sulla simulazione in atto:



Allocazione spazio su disco

Per la simulazione dell'allocazione dello spazio su disco da parte del File system si riutilizzerà OS Sim, selezionando *Help*, *Examples*, *File System Management* e scegliendo *Indexed Allocation (UNIX), indireccions*. Si tratta di un esempio che mostra il funzionamento dei file system ext2 e derivati.



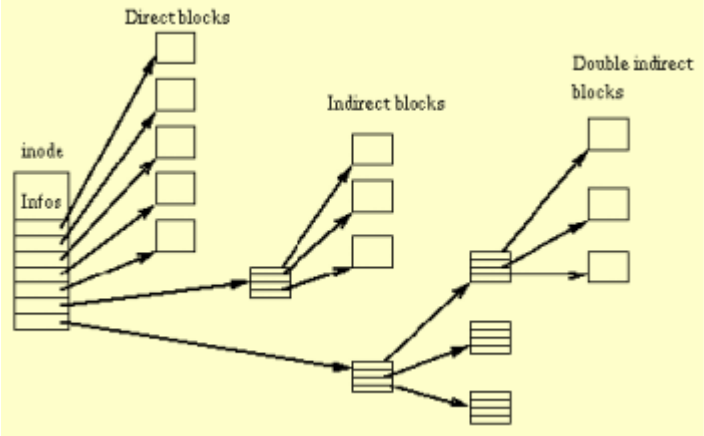
Nella parte alta della finestra non sono presenti pulsanti di avvio, sospensione, avvio step by step della simulazione perché qui si interagisce cancellando, generando nuovi file/cartelle e osservando la gestione dell'assegnazione di spazio.

Nella parte **1** della finestra è visualizzato il file system per come lo vede l'utente: un insieme di cartelle, file, sottocartelle. Il clic destro visualizza un menù da cui si può scegliere di aggiungere, cancellare o modificare le proprietà di un file/cartella.

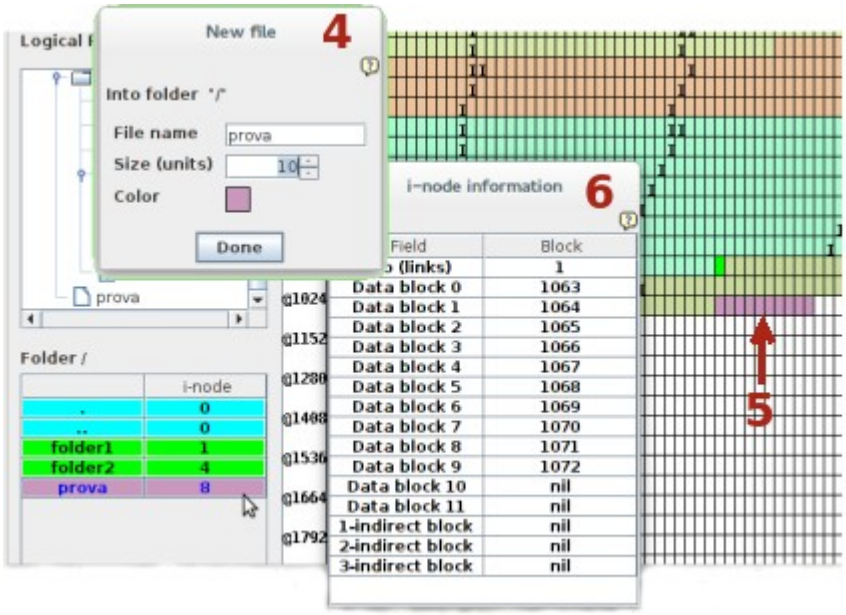
La zona **2** mostra il contenuto della cartella selezionata nella parte sovrastante. Qui il clic sinistro permette la visualizzazione delle informazioni relative all'i-node associato al file. Nella tabella dell'elenco sono presenti, anche, la riga che rappresenta la directory stessa (individuata dal punto) e quella che rappresenta la directory genitore (individuata dai due punti). Per esempio l'i-node della root si trova nel blocco 0 e se si seleziona, per esempio, la directory *folder1* si potrà notare che l'i-node della directory è allocato nel blocco 1 e la riga dedicata alla directory genitore punta al blocco 0 che è appunto il blocco contenete l'i-node della root.

La zona **3** mostra la memoria fisica divisa in blocchi con, sulla sinistra, l'indirizzo su disco. I primi 128 blocchi sono dedicati al Sistema Operativo, i blocchi allocati a un file sono dello stesso colore associato al file, le directory sono di colore verde, come in **2**, i blocchi segnati con *I*, *II* o *III* sono blocchi di indirezione, rispettivamente, di primo, secondo e terzo livello, i blocchi di colore bianco sono disponibili. Il pulsante *Data and statistics* mostra una tabella di tutti i blocchi, dal 128, con il relativo tipo (*Data block* o *indirect block*). In verde i blocchi dedicati alle cartelle.

Nella simulazione ogni i-node contiene 12 puntatori diretti a blocchi, 1 puntatore a un blocco di indirezione (il blocco può contenere fino a 12 puntatori a blocchi di dati), 1 puntatore a blocco di doppia indirezione e 1 puntatore a blocco di tripla indirezione in accordo con lo schema riprodotto a seguire e presente nell'help del programma:



I blocchi di indirezione sono utilizzati quando servono.

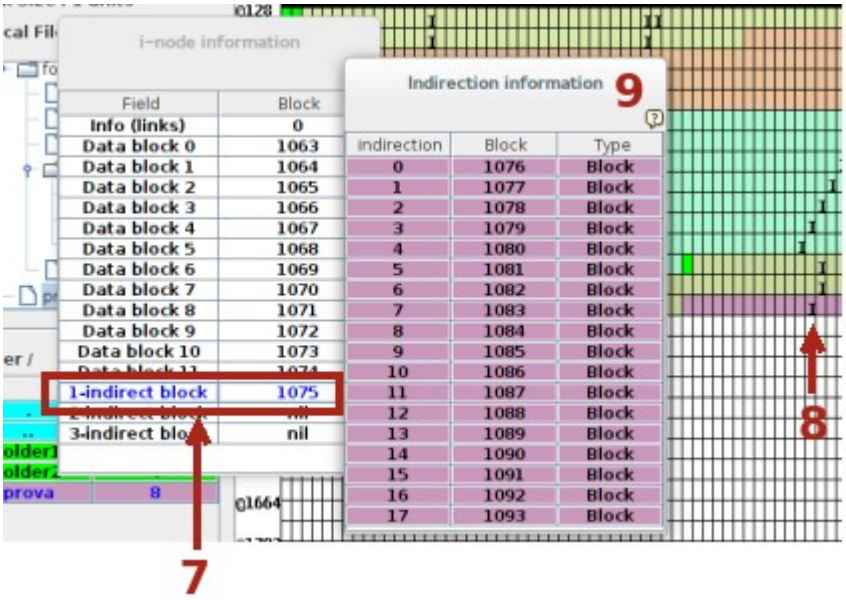


Selezionando la root e scegliendo, dal menù visualizzato nella parte della finestra con il File system logico con il clic destro del mouse, *Add file* si può generare un nuovo file (**4**) specificandone la dimensione in blocchi. Confermando l'operazione la memoria fisica (**5**) mostra i blocchi occupati con lo stesso colore scelto per il file. Selezionando con il tasto sinistro del mouse, dall'elenco dei file della cartella, il file generato si può esaminare l'i-node con allocati, in questo caso, i dieci blocchi dal 1063 al 1072 (**6**). I campi relativi ai puntatori agli ultimi due blocchi e i puntatori ai blocchi di indirezione, non utilizzati, hanno il valore *nil* nel puntatore. Lo spazio occupato dall'i-node è fisso ma, in questo momento, non sono allocati i blocchi di indirezione.

Non è importante che i blocchi siano contigui: nell'esempio dipende dal fatto che non si è cancellato alcun file di quelli esistenti e si è aggiunto, nello spazio disponibile, il nuovo file. L'accesso alle varie parti del file è garantito dalla presenza dei puntatori ai blocchi che ne garantiscono un accesso diretto.

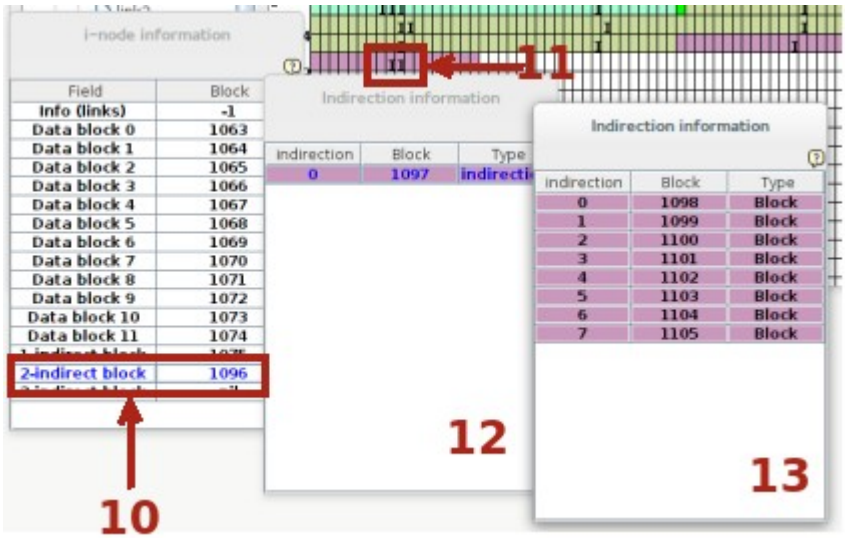
Mano a mano che il file necessita di ulteriore spazio, il File system per prima cosa utilizza i due puntatori ancora disponibili per assegnare due blocchi, ma se le richieste sono superiori si alloca lo spazio per il blocco dei puntatori indiretti che ne potrà contenere fino ad altri venti. Se dal File

system logico si rende disponibile il menù relativo la file *prova*, generato in precedenza, e si sceglie *Update* si può modificare il numero di unità richieste, portandolo per esempio a 30.



Il sistema, oltre ad utilizzare i due puntatori rimasti inutilizzati prima, alloca lo spazio per un blocco per l'indirezione (**7**). Nella memoria fisica tale blocco è evidenziato da *I* (**8**). Se si clicca sulla riga **7** si può esaminare l'i-node (**9**) con l'elenco dei blocchi assegnati.

Se le dimensioni del file aumentano ancora portando il numero di unità a 40 si ha la seguente situazione:



Viene allocato il blocco di doppia indirezione (**10**) evidenziato con *II* anche nella memoria fisica (**11**). cliccando sulla riga **10** viene visualizzato l'i-node (**12**) che punta al blocco 1097 che contiene (**13**) otto puntatori a blocchi. Se le dimensioni del file crescono ulteriormente si potrebbe avere bisogno del puntatore di tripla indirezione.

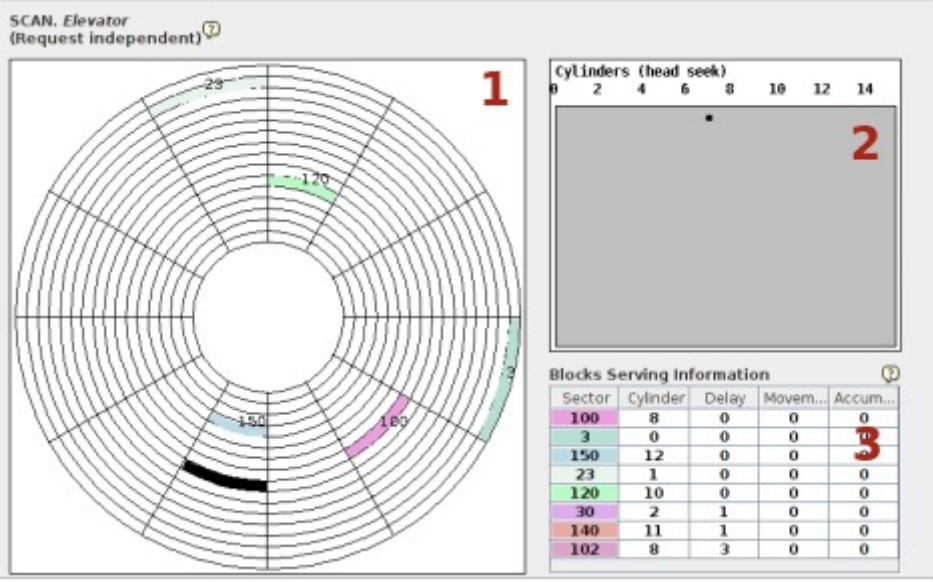
Potrebbe essere interessante osservare l'allocazione dello spazio in un file generato dopo averne cancellato qualche altro. I risultati saranno identici ma i blocchi allocati non saranno più contigui.

Scheduling richieste I/O

L'ultimo modulo trattato in questi appunti e simulato ancora dal software OS Sim riguarda gli accessi al disco. Si tratta di una parte importante e decisiva per le prestazioni generali di un sistema: i dischi sono più lenti rispetto alla memoria centrale (circa 1.000.000 volte più lenti) ma gli accessi ad essi sono molto frequenti vuoi per le richieste di un processo, vuoi per le richieste di swapping di pagine da parte del S.O.

Il tempo di accesso al settore contenente i dati è dipendente dal tempo di spostamento della testina sulla traccia (*tempo di seek*) e dal tempo occorrente per posizionarsi all'inizio del settore da leggere/scrivere (*tempo di rotazione*). In generale, quindi, un accesso sequenziale è più efficiente di un accesso random ma le richieste di accesso, specie in un sistema multitasking, avvengono in modo random. Le politiche di scheduling, al fine di diminuire i tempi necessari alle operazioni I/O intervenendo sui tempi di seek, devono poter trasformare sequenze di richieste di accessi random in sequenze ordinate.

Per poter osservare gli effetti delle diverse politiche di scheduling delle richieste si può selezionare *Help*, *Exercises*, *Disk management* e quindi *Algorithm comparison*.



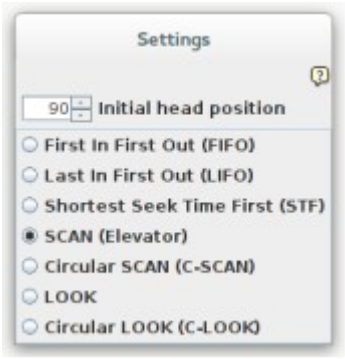
Nella finestra, come solito, nella parte sinistra in alto è specificato, in questo caso, l'algoritmo di scheduling utilizzato. Nella parte **1** è visualizzato il disco, unico, che è diviso in 16 cilindri, ogni cilindro è suddiviso in 12 settori per un totale di 192 settori dedicati ai dati. La posizione attuale della testina è indicata da un settore nero. I settori oggetto delle richieste di accesso vengono visualizzate con il relativo colore oltre che dal numero di settore.

Nella zona **2** viene visualizzato il grafico dello spostamento effettuato della testina per soddisfare le richieste. Nell'asse x è specificato il cilindro, nell'asse y il tempo trascorso. Ogni richiesta servita viene rappresentata da un punto, dello stesso colore della richiesta, con ascissa il numero di cilindro. Tutti i punti sono uniti da segmenti che indicano lo spostamento della testina.

La zona **3** è dedicata alle richieste. Per ognuna è presente una riga nella tabella comprendente: *Sector*, *Cylinder* (coordinate della richiesta), *Delay* (il tempo di arrivo nella lista delle richieste dall'inizio della simulazione), *Movement* (movimenti della testina che sono stati necessari per soddisfare la richiesta contati a partire dalla posizione precedente), *Accumulated* (totale spostamenti

che sono stati necessari per soddisfare la richiesta contati dall'inizio della simulazione). Il settore e il tempo di attesa della richiesta sono specificati nel momento della generazione della richiesta, il cilindro si deduce dalla prima informazione, gli ultimi due campi della riga sono riempiti in seguito al soddisfacimento della richiesta stessa.

Quando si inizia una nuova simulazione o quando si seleziona il pulsante *Settings* viene visualizzata una finestra in cui inserire i parametri della simulazione.



Nella finestra è possibile scegliere la posizione iniziale della testina e il tipo di politica da utilizzare. Per l'esempio scelto la testina, inizialmente, si trova nel settore 90 e la politica scelta è *SCAN*. Il simulatore permette di scegliere, per una comparazione, fra 7 politiche diverse. Tralasciando le prime due (*FIFO* e *LIFO*) che sono autoesplicative, si può scegliere in aggiunta fra *STF* (è selezionata la richiesta il cui settore è raggiungibile con il minor tempo di seek dalla posizione corrente), *SCAN* (si scansiona l'intera superficie del disco servendo le richieste che si incontrano mano a mano. Alla fine della scansione dei cilindri si inverte il senso), *LOOK* (variante della precedente: la testina si sposta fino alla posizione della richiesta più lontana e poi inverte il cammino senza arrivare alla fine). Le varianti circolari (*C-SCAN* e *C-LOOK*) si differenziano dalle varianti base perché quando la scansione arriva nella posizione più bassa, il cammino si inverte e la testina è portata nuovamente nel primo cilindro e durante il ritorno non vengono servite richieste.

Per avere una idea della differenza di prestazioni fra le varie politiche si possono avviare più simulazioni, sulle stesse richieste, cambiando ogni volta la politica. Quando si avvia la simulazione si nota che la posizione della testina si sposta in continuazione e quando è stata soddisfatta l'ultima richiesta si può premere il pulsante *pause* e osservare la tabella delle richieste.

Nella simulazione dell'esercizio la testina, al momento dell'avvio della simulazione, si trova nel settore 90 e ci sono 8 richieste (vedere **3**) di cui le ultime 3 vengono inserite con un ritardo di 1 unità le prime due e di 3 unità l'ultima.

FIFO

Blocks Serving Information

Sector	Cylinder	Delay	Movem...	Accum...
100	8	0	1	1
3	0	0	8	9
150	12	0	12	21
23	1	0	11	32
120	10	0	9	41
30	2	1	8	49
140	11	1	9	58
102	8	3	3	61

STF

Blocks Serving Information

Sector	Cylinder	Delay	Movem...	Accum...
100	8	0	1	1
120	10	0	2	3
140	11	1	1	4
150	12	0	1	5
102	8	3	4	9
30	2	1	6	15
23	1	0	1	16
3	0	0	1	17

L'algoritmo *FIFO*, come c'era da aspettarsi visto che le richieste interessano settori random, ha

prestazioni abbondantemente inferiori rispetto all'algoritmo *STF*: 61 spostamenti in totale per il primo contro i 17 del secondo. Nell'algoritmo *STF* le richieste non sono soddisfatte in ordine di arrivo ma lo spostamento delle testine fra una richiesta e l'altra (*Movement*) è minimo e gli spostamenti complessivi sono drasticamente inferiori.

SCAN

Blocks Serving Information

Sector	Cylinder	Delay	Movem...	Accum...
100	8	0	1	1
120	10	0	2	3
140	11	1	1	4
150	12	0	1	5
102	8	3	10	15
30	2	1	6	21
23	1	0	1	22
3	0	0	1	23

LOOK

Blocks Serving Information

Sector	Cylinder	Delay	Movem...	Accum...
100	8	0	1	1
120	10	0	2	3
140	11	1	1	4
150	12	0	1	5
102	8	3	4	9
30	2	1	6	15
23	1	0	1	16
3	0	0	1	17

Gli algoritmi *SCAN* e *LOOK* forniscono prestazioni comparabili (si confrontino i risultati su altre sequenze di richieste degli esempi proposti dal simulatore) anche se in questo caso c'è una piccola differenza.

Le conclusioni cui la letteratura del settore arriva si possono così sintetizzare:

1. Le performance dipendono dal numero e dal tipo di richieste.
2. Il tipo di richieste dipende da come è implementato il file system.
3. Una scelta ragionevole porta a considerare come scelte di default gli algoritmi *STF* e *LOOK*.