Università degli Studi di Messina

GUI based Process Management Tool

Daryn Calangi - Mariachiara Trifirò

Università degli studi di Messina

Presentation Overview

Descrizione del problema

L'applicazione propone un metodo di gestione intuitivo per ottenere delle prestazioni ottimali dal proprio computer. In particolare consente la terminazione di applicazioni e processi che compiono operazioni secondarie rispetto alle priorità dell'utente.

Obiettivi

La soluzione proposta consiste in un'applicazione GUI che permette di:

- Visualizzare in modo ordinato e intuitivo le applicazioni in esecuzione e i processi attivi.
- Terminare i processi con privilegi amministrativi per liberare risorse.
- Aggiornare dinamicamente le informazioni sul consumo di risorse per aiutare l'utente a prendere decisioni informate in tempo reale.

Strumenti utilizzati

Per implementare il progetto in esame è stato utilizzato Python come linguaggio di programmazione, in particolare anche le libreria *Tkinter* e *subprocess* rispettivamente per creare l'interfaccia grafica e lanciare processi **`bash`** tramite cui sono stati sfruttati dei comandi di monitoraggio di sistema.

Comandi utilizzati

Per ottenere le informazioni all'interno delle due schermate da fornire all'utente sono stati utilizzati rispettivamente i seguenti comandi:

- **`wmctrl`**
- **`top`**

Per compiere le operazioni effettive dell'applicazione sono stati sfruttati anche:

- **`kill`**
- **`renice`**

wmctrl

Il comando **`wmctrl`** (*Window Manager Control*) è stato utilizzato per interrogare il gestore delle finestre del sistema. Specificando l'opzione `-l` viene visualizzato l'elenco delle finestre gestite dal *window manager* e l'opzione `-p` viene visualizzato il PID del processo associato alla rispettiva finestra.

Esempio output wmctrl

```
mariachiara@Laptop-Mariachiara:~$ wmctrl -l -p
0x03000005  0 3500   Laptop-Mariachiara @dadda - Discord
0x0240000a  0 7630   Laptop-Mariachiara mariachiara@Laptop-Mariachiara: ~
0x04c0003e  0 8250   Laptop-Mariachiara Mozilla Firefox
0x02800004  0 8707   Laptop-Mariachiara Calcolatrice
```

Figure: Output wmctrl, pid inclusi

top

Il comando **`top`** permette la visualizzazione periodica dell'elenco ordinato di processi di sistema. Questi sono ordinati in modo predefinito in base alla percentuale di utilizzo della CPU. Inoltre, è stata utlizzata l'ozione `-b` per eseguire il comando in `batch mode`, ovvero non interattivo in modo da poter reindirizzare l'output all'interfaccia grafica.

Esempio output top

```
mariachiara@Laptop-Mariachiara:~$ top -b -n1 | head -n17
top - 10:25:44 up 36 min,  1 user,  load average: 3,74, 4,15, 3,70
Tasks: 306 total,   1 running, 305 sleeping,   0 stopped,   0 zombie
%Cpu(s): 44,4 us, 11,1 sy,  0,0 ni, 44,4 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
MiB Mem :   6865,9 total,    649,9 free,   4188,4 used,   2485,7 buff/cache
MiB Swap:    977,0 total,    974,5 free,      2,5 used.   2677,4 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   3641 mariach+  20   0 1146,3g 775136 265568 S 168,8  11,0  59:10.43 Discord
   3553 mariach+  20   0   32,8g 220408 125408 S  18,8   3,1   3:37.96 Discord
   2314 mariach+   9 -11   84572  17364   9224 S  12,5   0,2   0:27.49 pipewire
     60 root      20   0       0      0      0 D   6,2   0,0   0:05.33 kworker+
   2382 mariach+  20   0  803264 135164  75152 S   6,2   1,9   1:48.43 Xorg
   2656 mariach+  20   0 4790032 268308 132920 S   6,2   3,8   2:02.61 gnome-s+
   7630 mariach+  20   0  548680  50888  38628 S   6,2   0,7   0:04.25 gnome-t+
   9480 mariach+  20   0   11596   5344   3292 R   6,2   0,1   0:00.02 top
      1 root      20   0  168644  13340   9228 S   0,0   0,2   0:01.67 systemd
      2 root      20   0       0      0      0 S   0,0   0,0   0:00.00 kthreadd
```

Figure: Output top

kill

Il comando **`kill`** viene utilizzato per inviare segnali ai processi in esecuzione. Di seguito è rappresentato un esempio:

```
kill -9 <PID>
```

renice

Il comando **`renice`** viene utilizzato per modificare la priorità di esecuzione di uno o più processi già in esecuzione. Di seguito è presentato un esempio:

```
renice 10 -p <PID>
```

Interfaccia grafica

Sono state realizzate due schermate rispettivamente per garantire la gestione delle finestre aperte per gli utenti meno esperti e la gestione dei processi attivi sotto forma di impostazioni avanzate.

Gestione applicazioni

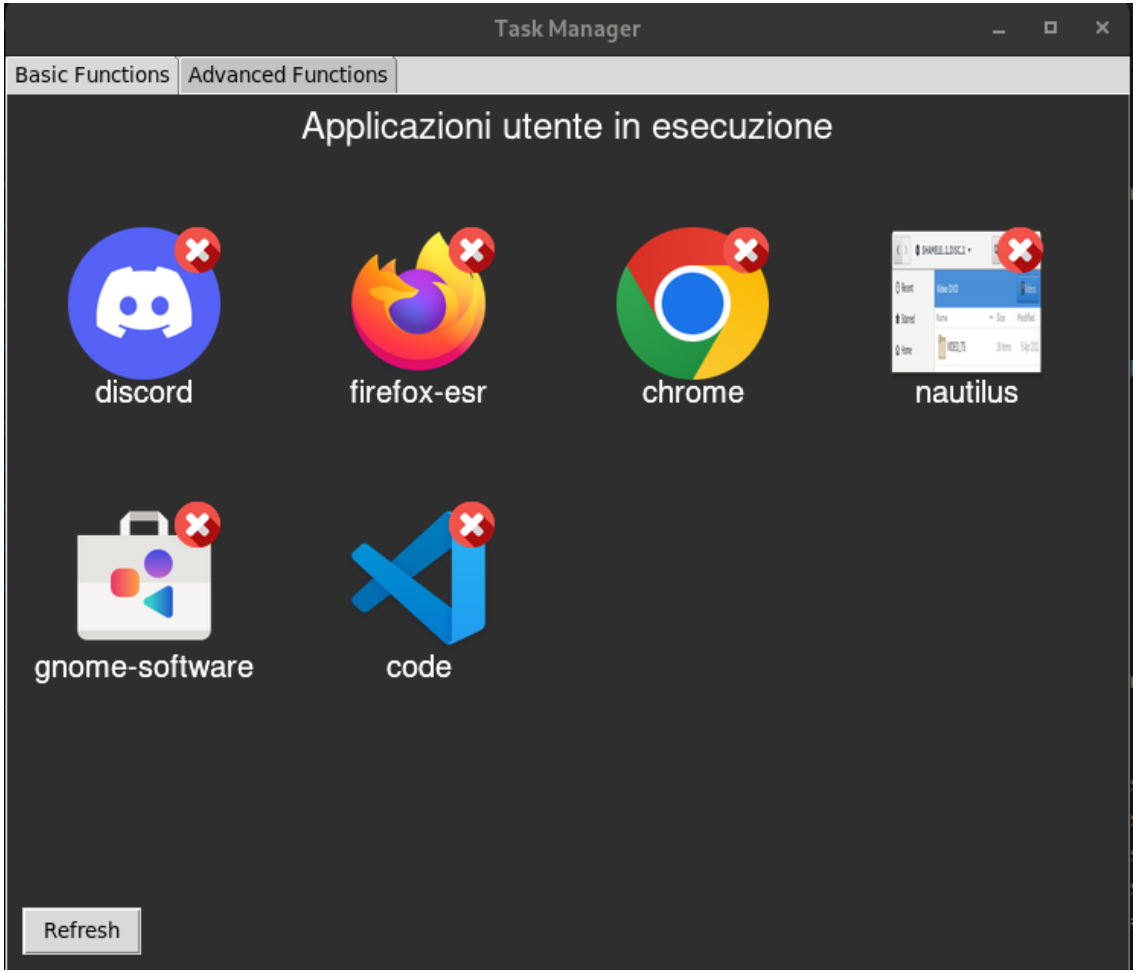


Figure: Schermata di gestione delle applicazioni

Gestione applicazioni

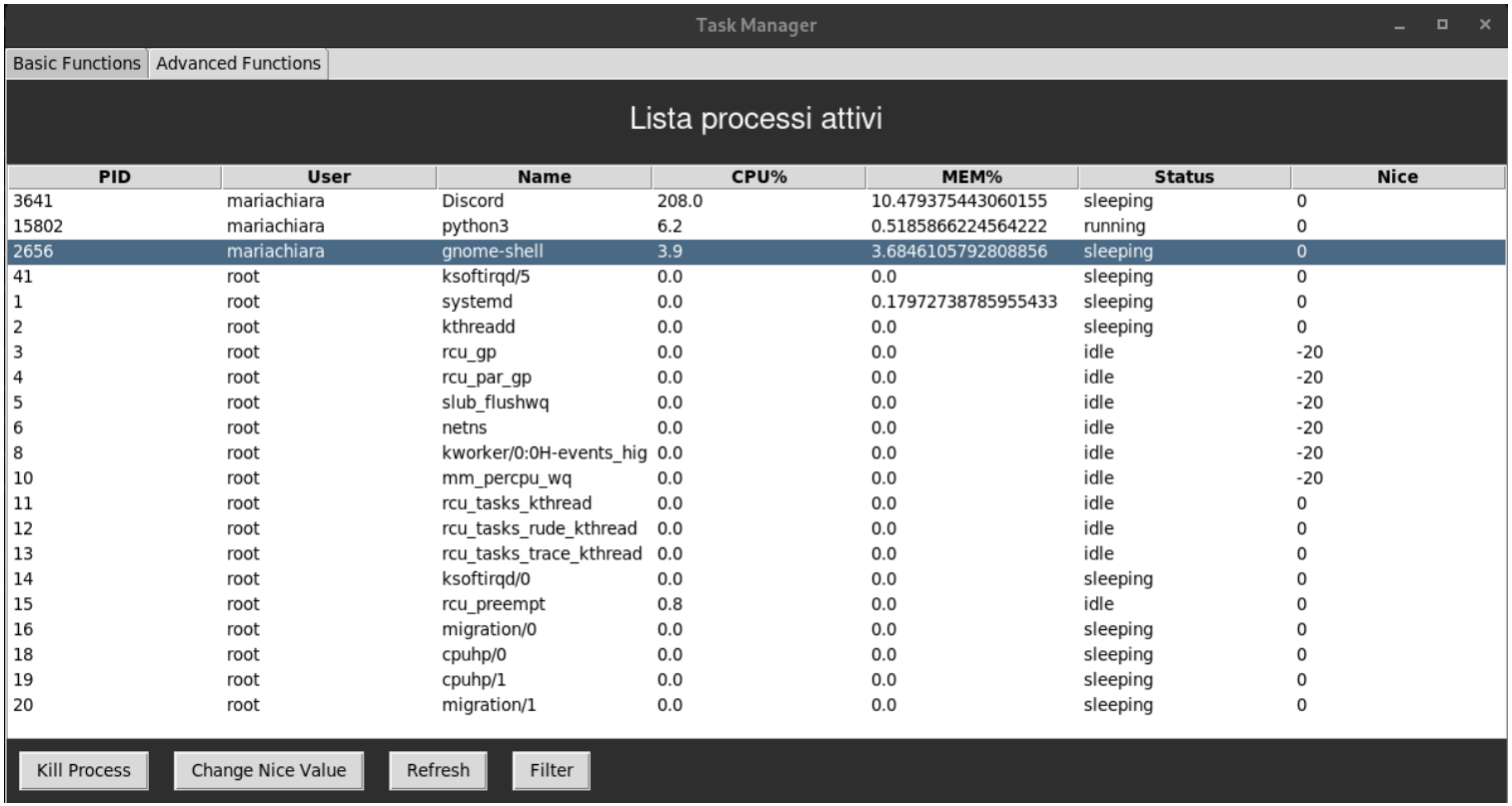
La prima interfaccia implementata garantisce una visualizzazione grafica delle applicazioni aperte, mostrando le rispettive icone reperite dinamicamente ricercandole all'interno del sistema stesso. Per ottenere le informazioni relative alle applicazioni è stato eseguito il comando **`wmctrl -l -p`**

Gestione applicazioni

Le operazioni concesse da questa schermata sono:

1. la visualizzazione delle applicazioni in esecuzione;
2. la terminazione delle applicazioni;
3. la riacquisizione delle applicazioni.

Gestione processi



PID	User	Name	CPU%	MEM%	Status	Nice
3641	mariachiara	Discord	208.0	10.479375443060155	sleeping	0
15802	mariachiara	python3	6.2	0.5185866224564222	running	0
2656	mariachiara	gnome-shell	3.9	3.6846105792808856	sleeping	0
41	root	ksoftirqd/5	0.0	0.0	sleeping	0
1	root	systemd	0.0	0.17972738785955433	sleeping	0
2	root	kthreadd	0.0	0.0	sleeping	0
3	root	rcu_gp	0.0	0.0	idle	-20
4	root	rcu_par_gp	0.0	0.0	idle	-20
5	root	slub_flushwq	0.0	0.0	idle	-20
6	root	netns	0.0	0.0	idle	-20
8	root	kworker/0:0H-events_hig	0.0	0.0	idle	-20
10	root	mm_percpu_wq	0.0	0.0	idle	-20
11	root	rcu_tasks_kthread	0.0	0.0	idle	0
12	root	rcu_tasks_rude_kthread	0.0	0.0	idle	0
13	root	rcu_tasks_trace_kthread	0.0	0.0	idle	0
14	root	ksoftirqd/0	0.0	0.0	sleeping	0
15	root	rcu_preempt	0.8	0.0	idle	0
16	root	migration/0	0.0	0.0	sleeping	0
18	root	cpuhp/0	0.0	0.0	sleeping	0
19	root	cpuhp/1	0.0	0.0	sleeping	0
20	root	migration/1	0.0	0.0	sleeping	0

Figure: Schermata di gestione dei processi

Gestione processi

Le operazioni concesse da questa schermata sono:

1. la visualizzazione dei processi;
2. la terminazione dei processi;
3. la modifica del valore di *nice*;
4. il filtraggio dei processi in base a vari criteri;
5. la riacquisizione dei processi.

Gestione processi

All'interno della finestra è mostrata una tabella contenente le colonne:

- **PID**, l'identificativo del processo;
- **User**, l'utente che ha lanciato il processo;
- **Name**, il nome associato alla finestra;
- **CPU%**, la quantità di CPU utilizzata dal processo;
- **MEM%**, la quantità di memoria utilizzata dal processo;
- **Status**, lo stato del processo;
- **Nice**, il valore di nice del processo.

Gestione processi

Sono stati inseriti i seguenti pulsanti:

1. **Kill Process** per terminare i processi;
2. **Change Nice Value** per modificare il valore di nice dei processi;
3. **Refresh** per aggiornare la lista;
4. **Filter** per applicare filtri.

Gestione processi

Per evitare eventuali errori dovuti alla terminazione di processi vitali per il corretto funzionamento del sistema è consentita la terminazione dei soli processi appartenenti all'utente corrente che non può corrispondere con l'utente *root*.

Filtro processi

Questa operazione può essere compiuta inserendo come primo elemento il campo d'interesse e come secondo il valore in base a cui filtrare, separati dai due punti. Un esempio di input che consente di filtrare i processi in base al nome è:

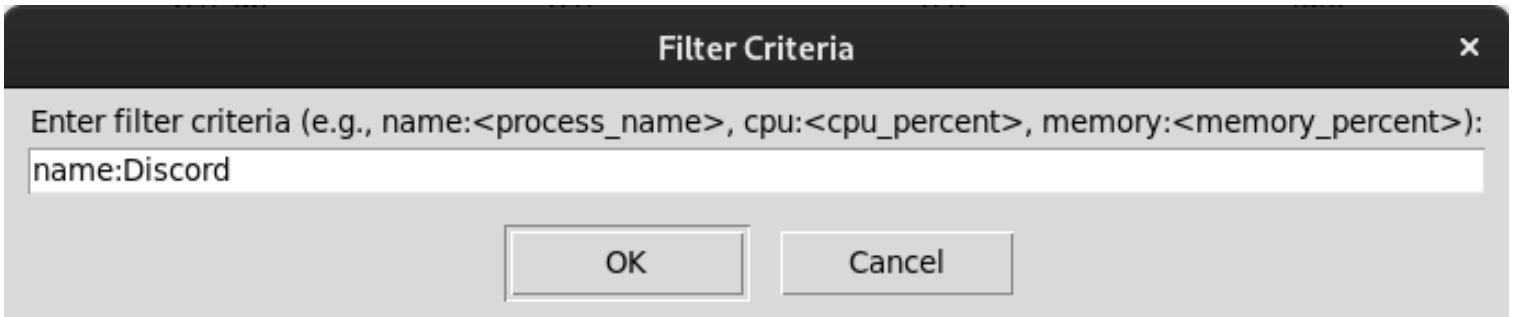


Figure: Finestra per aggiungere un filtro

Modifica valore di nice

Un ulteriore operazione consentita è la modifica del valore di *nice* di un processo appartenente all'utente che ha eseguito il codice. La finestra di dialogo che consente questa operazione è la seguente:

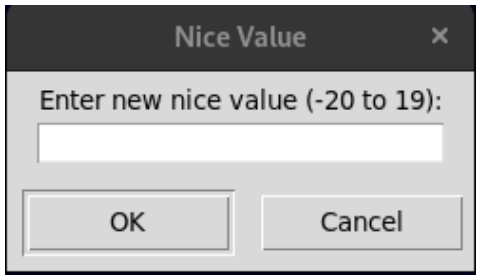


Figure: Finestra per modificare il valore di nice