

Marwan Chtini (VR510527)

Angelo Vaccaro (VR480616)

Questo programma gestisce due algoritmi di pianificazione dei task e include una serie di operazioni per la gestione dell'input dell'utente, la visualizzazione dei risultati e il calcolo della penalità. La sua struttura è divisa in due sezioni principali: `.data` e `.text`. Ecco una panoramica dettagliata del codice:

#### Sezione `.data`

Questa sezione definisce e inizializza variabili e stringhe usate dal programma. Ecco un riepilogo delle sue principali componenti:

- **Buffer e variabili di lunghezza:**
  - `buffer`: Spazio di 1024 byte per l'input dell'utente.
  - `buffer_len`: Lunghezza del buffer calcolata come differenza tra l'indirizzo corrente e `buffer`.
  - `bufferittoa`: Spazio di 1024 byte per la conversione di numeri in stringhe.
- **Menu e messaggi:**
  - `menu`: Stringa contenente il testo del menu di scelta dell'utente.
  - `scelta`: Stringa predefinita con il valore "0".
  - `numero_nv`: Messaggio di errore per input non valido.
  - `punti`, `ac`, `con`, `pen`, `pedf`, `phpf`: Messaggi vari usati durante l'esecuzione del programma, come il risultato dell'algoritmo di pianificazione e le penalità.
- **Variabili di lavoro:**
  - `tmp`, `contatore`: Variabili usate per il conteggio e altre operazioni temporanee.
  - Variabili `idx`, `durataX`, `scadenzaX`, `prioritaX`: Strutture dati per memorizzare le informazioni sui task (fino a 10 task).

#### Sezione `.text`

Questa sezione contiene il codice eseguibile del programma. Di seguito, è descritta in dettaglio:

1. **Avvio del programma (`_start`):**
  - Il programma inizia con la lettura dell'input dell'utente usando le syscall `read` e `write`.
  - Il buffer di input viene letto e la lunghezza dell'input viene salvata.
2. **Parsing dell'input:**
  - L'input viene analizzato per estrarre numeri separati da virgole e nuove righe.
  - I numeri estratti vengono memorizzati nelle variabili `idx`, `durataX`, `scadenzaX`, `prioritaX`.
3. **Visualizzazione del menu:**
  - Il programma visualizza il menu e attende l'input dell'utente.
4. **Gestione della scelta dell'utente:**
  - La scelta dell'utente viene convertita da stringa a numero.
  - A seconda della scelta (1 per EDF, 2 per HPF, 3 per uscire), il programma esegue l'algoritmo di pianificazione corrispondente o termina.
5. **Algoritmo EDF (Earliest Deadline First):**
  - I task vengono ordinati in base alla loro scadenza (deadline) in ordine crescente.
  - Viene visualizzato il risultato della pianificazione EDF.
6. **Algoritmo HPF (Highest Priority First):**
  - I task vengono ordinati in base alla loro priorità in ordine decrescente.
  - Viene visualizzato il risultato della pianificazione HPF.

## 7. Calcolo e visualizzazione della penalità:

- Dopo aver eseguito l'algoritmo scelto, il programma calcola la penalità associata ai task non completati entro la loro scadenza.
- Viene visualizzato il risultato della penalità.

## 8. Pulizia e terminazione:

- Vengono azzerati i buffer e le variabili.
- Il programma termina eseguendo la syscall `exit`.

### Dettagli Aggiuntivi

- **Conversione di numeri in stringhe (`itoa` e varianti):**

- Il programma include routine per convertire numeri interi in stringhe di caratteri per la visualizzazione.

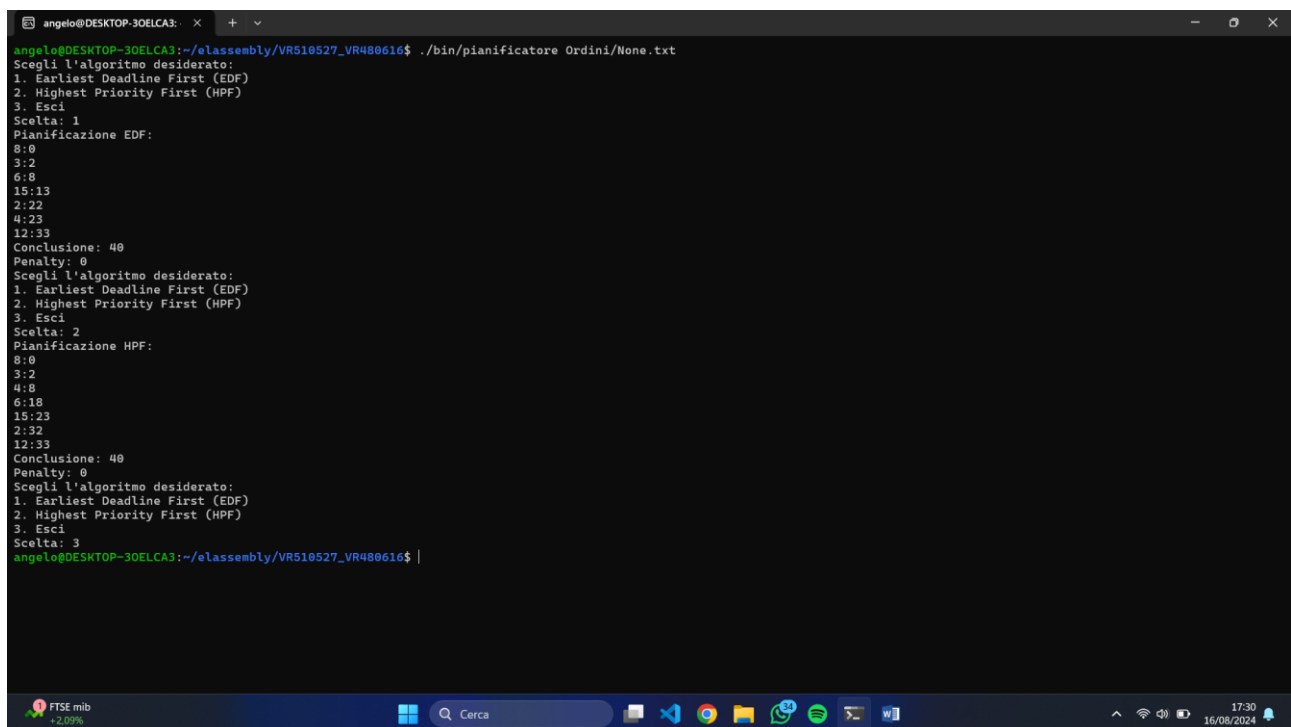
- **Gestione delle penalità:**

- La penalità viene calcolata come il prodotto tra il ritardo e la durata del task e viene visualizzata dopo la pianificazione.

### Conclusioni

Il codice fornito è un programma di pianificazione dei task che gestisce l'input dell'utente, esegue algoritmi di pianificazione (EDF e HPF) e calcola la penalità per i task non completati in tempo. Include anche la gestione dell'input, la visualizzazione dei risultati e la pulizia delle variabili. La struttura del programma è progettata per essere eseguita in ambiente Linux x86 e utilizza syscall per le operazioni di I/O.

### Output con il file di testo None.txt



```
angelo@DESKTOP-3OELCA3: ~$ ./bin/pianificatore Ordini/None.txt
Scegli l'algoritmo desiderato:
1. Earliest Deadline First (EDF)
2. Highest Priority First (HPF)
3. Esci
Scelta: 1
Pianificazione EDF:
8:0
3:2
6:8
15:13
2:22
4:23
12:33
Conclusione: 40
Penalty: 0
Scegli l'algoritmo desiderato:
1. Earliest Deadline First (EDF)
2. Highest Priority First (HPF)
3. Esci
Scelta: 2
Pianificazione HPF:
8:0
3:2
4:8
6:18
15:23
2:32
12:33
Conclusione: 40
Penalty: 0
Scegli l'algoritmo desiderato:
1. Earliest Deadline First (EDF)
2. Highest Priority First (HPF)
3. Esci
Scelta: 3
angelo@DESKTOP-3OELCA3: ~$
```

## Output con il file di testo EDF.txt

```
angelo@DESKTOP-3OELCA3: ~/elassembly/VR510527_VR480616$ ./bin/pianificatore Ordini/EDF.txt
Scegli l'algoritmo desiderato:
1. Earliest Deadline First (EDF)
2. Highest Priority First (HPF)
3. Esci
Scelta: 1
Pianificazione EDF:
8:0
11:2
20:6
3:8
6:14
9:19
2:22
16:23
12:32
4:39
Conclusione: 49
Penalty: 0
Scegli l'algoritmo desiderato:
1. Earliest Deadline First (EDF)
2. Highest Priority First (HPF)
3. Esci
Scelta: 2
Pianificazione HPF:
11:0
3:4
4:10
6:20
9:25
20:28
16:30
8:39
2:41
12:42
Conclusione: 49
Penalty: 136
Scegli l'algoritmo desiderato:
1. Earliest Deadline First (EDF)
2. Highest Priority First (HPF)
3. Esci
Scelta: 3
angelo@DESKTOP-3OELCA3: ~/elassembly/VR510527_VR480616$
```

## Output con il file di testo Both.txt

```
angelo@DESKTOP-3OELCA3: ~/elassembly/VR510527_VR480616$ bin/pianificatore Ordini/Both.txt
Scegli l'algoritmo desiderato:
1. Earliest Deadline First (EDF)
2. Highest Priority First (HPF)
3. Esci
Scelta: 1
Pianificazione EDF:
21:0
3:2
4:8
13:18
11:21
6:25
18:30
1:39
2:46
5:47
Conclusione: 49
Penalty: 8
Scegli l'algoritmo desiderato:
1. Earliest Deadline First (EDF)
2. Highest Priority First (HPF)
3. Esci
Scelta: 2
Pianificazione HPF:
21:0
3:2
4:8
11:18
13:22
6:25
18:30
5:39
1:41
2:48
Conclusione: 49
Penalty: 8
Scegli l'algoritmo desiderato:
1. Earliest Deadline First (EDF)
2. Highest Priority First (HPF)
3. Esci
Scelta: 3
angelo@DESKTOP-3OELCA3: ~/elassembly/VR510527_VR480616$
```