Laboratorio di Architettura degli Elaboratori
Elaborato Assembly 2024
Pianificatore ordini

GIORGIA ZANINI - VR503052

VADIM MUSTEATA - VR422249

A.A. 2023/2024

Indice

Capitolo 1

Specifiche

Il progetto realizzato prevede la realizzazione di un software per la pianificazione delle attività di un sistema produttivo.

Esso prevede la pianificazione fino a un massimo di 10 prodotti, mandati in produzione uno dopo l'altro, quindi bisogna aspettare la fine della produzione dell'ordine precedente per far iniziare il successivo. Questi sono distribuiti in un massimo di 100 unità di tempo, chiamati "slot temporali".
Ogni ogni ordine è caratterizzato da 4 valori:

1. **Identificativo**: è l'identificativo dell'ordine stesso. Può avere un valore compreso tra 1 e 127.
2. **Durata**: è il tempo che serve al prodotto per essere completato. Espresso in unità di tempo. Può avere un valore compreso tra 1 e 10, il che fa si che non si possano superare un massimo di 100 unità di tempo.
3. **Scadenza**: è l'unità di tempo massima in cui terminare la produzione del prodotto stesso. Può avere un valore compreso tra 1 e 100.
   Nel caso in cui questo venisse superato, la produzione subisce una penalità per ogni prodotto che supera la sua rispettiva scadenza, sommate poi per dare la penalità totale.
4. **Priorità**: Può avere un valore compreso tra 1 e 5, dove 5 indica la priorità massima e un la minima.

Gli ordini sono salvati in un file di testo txt, dove i valori di ogni prodotto sono separati da una virgola, e ogni riga corrisponde a un ordine.

Penalità

Per ogni prodotto completato in ritardo rispetto alla scadenza indicata, l'azienda dovrà pagare una penale.

Viene quindi calcolata una penale per ogni prodotto che finisce oltre la scadenza:

$$Penalty = Priority * (tempo\_fine - Scadenza)$$

Vengono poi sommate le penalità di ogni prodotto, se ne ha, per ottenere la penalità complessiva.

$$Penalty = \sum_{i=1}^{n} Priority * (tempo\_fine - Scadenza)$$

Capitolo 2

Struttura del codice e scelte progettuali

2.1 Descrizione generale del codice

Il programma parte col *main* che prende in input uno o due file e li apre , in seguito fa una chiamata alla funzione *salva_numeri* che salva i numeri del file in un array per poi ritornare nel *main*.

Poi viene chiamato il *menu*, che "prende il controllo", qui viene stampata a video la scelta tra la pianificazione *"EDF"* o la pianificazione *"HPF"*, o *"Esci"*. In base alla scelta fatta avverrà una chiamata alla funzione *ordinamento_EDF*, che farà l'ordinamento in base all'algoritmo EDF (Earliest Deadline First) e successivamente farà la chiamata alla funzione *elabora_ordini*, *ordinamento_HPF*, che farà l'ordinamento stavolta in base all'algoritmo HPF (Highest Priority First) e chiamerà anch'essa la funzione *elabora_ordini*, o *termina_programma*, che chiuderà i file aperti e terminerà il programma.

elabora_ordini invece si occupa di effettuare la pianificazione effettiva, calcolandone anche il tempo e un'eventuale penalità, per poi stampare a video e, in base ai parametri passati, anche nel file, per poi ritornare al *menu*.

Il programma ritornerà sempre nel *menu* finché non verrà scelta l'opzione *"Esci"*.

2.2 Descrizione *main* e funzioni

2.2.1 main

Il *main* ha il compito di analizzare i parametri passati da linea di comando e aprendo, se presenti, i file forniti.
Il *main* esegue dalle 3 alle 4 istruzioni pop:

- La prima istruzione di pop preleva dallo stack il numero di parametri passati da terminale e lo salva nella variabile *counter_parametri*;
- La seconda preleva il nome del programma, che non viene utilizzato;

- La terza preleva il percorso del file degli ordini;
- Il quarto preleva il percorso del file per salvare i risultati.

Prelevato il percorso del file degli ordini, viene verificato che questo non sia vuoto, in caso prova ad aprirlo con la system call open in sola lettura.
Se il risultato restituito è negativo il programma termina con un errore relativo all'apertura del file.
Se la open ha successo viene verificato il numero salvato nella variabile *counter_parametri*: se il suo valore è uguale a 3 il programma suppone che sia stato fornito in input il percorso del file per salvare i risultati, esegue quindi solo in questo caso la quarta pop, verifica che il valore estratto non sia vuoto, per poi lanciare una nuova system call open provando ad aprire questo file in sola scrittura.
Anche qui se il risultato è negativo il programma termina con un errore relativo all'apertura del file.
Aperto con successo il file degli ordini (e il file per salvare i risultati), il *main* richiama la funzione *salva_numeri*, e al termine di questa, la funzione *menu*.

2.2.2 salva_numeri

La funzione *salva_numeri*, dato in input il file descriptor associato al file degli ordini, ha il compito di leggere il file e salvarne il contenuto in un array, di dimensione 40.
La funzione legge un carattere per volta, e se questo è un numero, lo salva in un array temporaneo, di dimensione 4, per contenere fino a tre cifre e il terminatore per renderlo una stringa; nel momento in cui viene letto un carattere diverso, che può essere una virgola o un a capo, va a convertire la stringa in un intero, esegue i controlli per verificare che il numero rientri nei parametri previsti (se corrisponde all'id del prodotto, verifica che sia compreso tra 1 e 127, ...), per poi salvarlo nell'array principale.
Terminata le lettura del file e salvati tutti i numeri nell'array degli ordini, la funzione verifica che il numero di ordini non sia 0, non sia a 10, e che ciascun ordine sia composto da 4 valori.
Superati i controlli, la funzione restituisce il puntatore all'array degli ordini e la dimensione di quest'ultimo (numero di ordini * 4).
In caso di errore, che sia relativo al range del valore di un ordine di un valore letto o al numero degli ordini, stampa l'errore relativo e termina.

2.2.3 menu

La funzione *menu*, prende in input il file descriptor del primo e del secondo file, il puntatore all'array degli ordini e la dimensione di quest'ultimo, e fornisce la scelta tra i due possibili ordinamenti e l'uscita dal programma.
Nel caso degli ordinamenti, rimanda alla funzione rispettiva all'ordinamento scelto, e una volta terminata l'esecuzione di tale ordinamento e la conseguente pianificazione dell'ordine, ritornerà nella scelta proposta dal *menu*; mentre con l'opzione "Esci", essa chiamerà la funzione *termina_programma*, per terminare il programma.

2.2.4 ordinamento_EDF

Prende in input il puntatore all'array degli ordini e la rispettiva dimensione, e ordina l'array in base alla scadenza (più è vicina, prima verrà prodotto), quindi gli ordini verranno disposti in ordine crescente rispetto al terzo numero di ognuno. Nel caso in cui due ordini abbiano la medesima scadenza, verrà controllata la priorità di quella coppia, ovvero il quarto numero dei due gruppi (in questo caso la disposizione sarà decrescente, guardando il valore della priorità), per poi ritornare all'ordinamento principale della funzione. Se, sia la scadenza, che la priorità sono uguali, i due ordini confrontai rimangono nella disposizione di partenza.
La funzione utilizza il metodo del bubble sort per eseguire l'ordinamento.

2.2.5 ordinamento_HPF

Prende in input il puntatore all'array degli ordini e la rispettiva dimensione, e ordina l'array in base alla priorità (più è alta, prima verrà prodotto), quindi gli ordini verranno disposti in ordine decrescente rispetto al quarto numero di ognuno. Nel caso in cui due ordini abbiano la medesima priorità, verrà controllata la scadenza di quella coppia, ovvero il terzo numero dei due gruppi (in questo caso la disposizione sarà crescente, guardando il valore della scadenza), per poi ritornare all'ordinamento principale della funzione. Se, sia la priorità, che la scadenza sono uguali, i due ordini confrontai rimangono nella disposizione di partenza.
La funzione utilizza il metodo del bubble sort per eseguire l'ordinamento.

2.2.6 elabora_ordini

Prende in input il file descriptor del file per salvare i risultati, il puntatore all'array degli ordini e la sua dimensione.
Per ogni ordine stampa a video e su file (se fornito) l'id del prodotto (primo valore) e il relativo tempo di inizio produzione, che equivale al tempo di fine del prodotto precedente (o al tempo 0 se si tratta del primo prodotto). In seguito calcola il tempo che ci impiega il prodotto ora in produzione, tramite la durata (secondo valore), lo salva in memoria per il tempo di inizio del prodotto successivo e lo confronta con la scadenza (terzo valore) per sapere se nell'ordine dato c'è o meno penalità, e in caso calcolarla.
Al termine stampa inoltre il tempo di fine e la penalità ottenuta.

2.2.7 converti_str_a_int

Prende in input la stringa da convertire.
Partendo dall'inizio della stringa, prende una cifra, che va poi a sommare al numero salvato finora (inizializzato a 0) moltiplicato per 10 (*numero_salvato = (numero_salvato * 10) + nuova_cifra*).
Esegue questo loop fino a quando trova un terminatore.
Ritorna poi l'intero ottenuto.

2.2.8 converti_int_a_str

La funzione prende in input un intero.
Scompone il valore numerico passatogli, prendendo l'ultima cifra e convertendola in ascii, per poi salvarla in un array, partendo dalla penultima cella. Esegue questo loop finché non saranno salvate tutte le cifre.
Nell'ultima cella viene poi posto un terminatore.
Ritornerà poi il puntatore dell'array alla cella corrente, che corrisponderà all'inizio della stringa.

2.2.9 stampa_stringa

Questa funzione prende in input il puntatore a una stringa da stampare e il file descriptor associato al file per salvare i risultati. Se questo è diverso da -1 la stampa avviene anche nel file, altrimenti solo su terminale.
Dato il puntatore alla stringa, che deve sempre terminare con \0, la funzione conta il numero di caratteri che la compongono (escluso il terminatore) e chiama poi la system call write, fornendo come dimensione da stampare il numero contato.

2.2.10 termina_programma

Questa funzione prende in input i file descriptor associati ai due file passati da linea di comando, li chiude se aperti e termina l'esecuzione.
Un file viene considerato aperto se il valore del suo file descriptor è maggiore di -1.

2.2.11 stampa_array

(funzione di debug) Prende in input il puntatore all'array e la sua dimensione, e, tramite *converti_int_a_str* e *stampa_stringa*, converte i valori dell'array e li stampa solo su terminale, intervallati da spazi.

2.3 Disposizione file

- VR503052_VR422249/
  - src/
    - converti_int_a_str.s
    - converti_str_a_int.s
    - elabora_ordini.s
    - main.s
    - menu.s
    - ordinamento_EDF.s
    - ordinamento_HPF.s
    - salva_numeri.s
    - stampa_array.s
    - stampa_stringa.s

    - termina_programma.s
  - obj/
  - bin/
  - Ordini/
    - EDF.txt
    - HPF.txt
    - Both.txt
    - None.txt
  - Makefile
  - Relazione.pdf

In src sono presenti i file del *menu* e di tutte le funzioni, invece obj sarà la destinazione di tutti i file compilati, mentre bin conterrà l'eseguile "pianificatore".
Per creare i file che verranno contenuti in obj e bin, è stato creato un Makefile da lanciare, che si occupa anche della pulizia delle stesse cartelle.
Mentre i file di input e di destinazione sono contenuti dentro Ordini; i tipi di file sorgente possono essere:

- EDF.txt, che ha penalità uguale a zero con EDF e maggiore di zero con HPF
- HPF.txt, che ha penalità uguale a zero con HPF e maggiore di zero con EDF
- Both.txt, che ha penalità uguale a zero con entrambi gli algoritmi
- None.txt, che ha penalità maggiore di zero con entrambi gli algoritmi

2.4 Scelte progettuali

1. **Apertura e lettura file**
   Il programma apre i file una volta sola nel *main*, una volta aperti salva i numeri in un array grande al massimo 10, e utilizzerà sempre quell'array, ordinandolo in base alla necessità, perché i file e gli ordini resteranno sempre gli stessi fino all'uscita del programma, l'unica cosa che cambierà e il tipo di ordinamento, che vien gestito chiamando l'ordinamento già dal *menu*.
   Quindi una volta acquisiti gli ordini sarà il *menu* a gestire quale tipo di ordinamento effettuare, per poi passare all'elaborazione dell'ordine e ritornare nel *menu* stesso; ed è sempre il *menu* che si occuperà della chiusura del programma.

2. **Opzione *"Esci"* nel *menu***
   Nel *menu* sono presenti 3 possibili scelte, le prime due sono i 2 tipi di ordinamento previsti, mentre la terza esce dal programma, affinché all'uscita vengano anche chiusi i file aperti.

3. **Ordinamento nell'array stesso**
   Nell'ordinamento viene utilizzato un unico array, ovvero quello passatogli, nel quale, per ordinarlo, vengono scambiati gli ordini a due a due, tramite l'utilizzo dello stack, utilizzato con la funzionalità di variabile temporanea.

4. **Utilizzo dei registri**
   Nelle funzioni per passare i diversi parametri, invece dello stack, vengono utilizzati i registri:

   - EAX: per il file descriptor del file degli ordini;
   - EBX: per il file descriptor del file per salvare i risultati;
   - ESI: per il puntatore all'array degli ordini
   - ECX: per la dimensione dell'array degli ordini

   fatta eccezione per le funzioni *stampa_stringa*, *converti_str_a_int*, *converti_int_a_str*, *stampa_array*, in cui EAX viene utilizzato per passare l'intero o il puntatore alla stringa o all'array.

5. **Scrittura su terminale e file**
   Per la scrittura su terminale e su file è stata creata un'unica funzione generica di stampa. Essa esegue la system all write prima su terminale, e se il valore del file descriptor passatogli è diverso da -1 allora la esegue anche su file. Ciò significa che se il valore del file descriptor è -1, o il file non è stato fornito, o è stato passato al posto del file descriptor il valore -1, in modo stampare solo su terminale.

6. **Terminazione programma**
   Il programma non termina nel *main*, ma termina grazie alla chiamata a un'altra funzione, chiamata *termina_programma*, che si occupa di chiudere i file se aperti e terminare il programma stesso, in questo modo la chiusura del programma potrà essere chiamata in qualsiasi momento, sia per la chiusura effettiva data dalla scelta "Esci" presente nel *menu*, sia al verificarsi di un qualsiasi errore.

7. **Passaggio da uno a due file tramite comando da terminale**
   Eseguendo il codice si può decidere si può decidere se passare in input un solo parametro, che corrisponde al file degli ordini, inserendo la seguente riga di comando da terminale:
   `./bin/pianificatore <percorso del file degli ordini>`
   oppure si può decidere di inserire come parametro passato, oltre al file degli ordini, un file di destinazione, dove viene salvato l'algoritmo usato e la pianificazione degli ordini in base all'algoritmo scelto.
   `./bin/pianificatore <percorso del file degli ordini> <percorso del file di pianificazione>`