SISTEMI OPERATIVI IIN

SISTEMI DI ELABORAZIONE P.O.
prova scritta preliminare del 18.03.2004

Nome: _____ Cognome: _____

In un sistema Linux/Unix è in esecuzione un processo P_0 che fornisce un servizio. Il processo riceve le richieste attraverso una chiamata alla procedura bloccante `Object get_next_request()`, che restituisce un generico dato di tipo `Object` che deve essere elaborato.
Il processo P_0 rimane continuamente in ascolto per le richieste e per ognuna genera un nuovo processo lavoratore P_i a cui deve passare l'oggetto da elaborare.
I processi lavoratori eseguono l'elaborazione dell'oggetto attraverso la chiamata alla procedura `void perform_work(Object)`, che restituisce il risultato della elaborazione. In aggiunta, i processi P_i devono scrivere, ognuno su un proprio file, un numero che li identifica in ordine di creazione, ed il risultato della elaborazione, quest'ultimo utilizzando la procedura `void print_object(FILE *, Object)`.

Si delinei la possibile struttura del processo P_0.

Soluzione.

Codice del processo P_0 in pseudo-codice C.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

typedef struct {
        …..
} Object;

int main ()
{
        Object data;
        int pid;
        int number = 0;       // numero del figlio

        while (1) {
                data = get_next_request();
                number ++;

                pid = fork();
                if (pid<0) {
                        // errore esecuzione della fork
                        printf("\nImpossibile generare processo figlio");
                        return 0;
                }
                else if (pid>0) {
                        // codice eseguito dal processo padre
                }
                else {
                        // codice eseguito dal processo figlio
                        FILE * fp;
                        char name[50];

                        perform_work(data);

                        sprintf(name,"%d.txt",number);
                        fp = fopen(name);
                        fprintf(fp,"%s",name);
                        print_object(fp,data);
                        fclose(fp);
                }
        }

}
```