Esempio 2

martedì 16 maggio 2023 15:22

Andiamo a vedere un esempio in cui andiamo a mappare della memoria condivisa.

```c
#include <stdlib.h>
#include <stdio.h>
#include <sys/mman.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>

#define PAGE_SIZE (4096)

#define NUM_TARGET_PAGES 10000

int gets(char*);

int main(int argc, char** arv){

    pid_t pid;
    char* buffer;


    buffer = (char*)mmap(NULL,PAGE_SIZE*NUM_TARGET_PAGES,PROT_READ|PROT_WRITE,MAP_ANONYMOUS|MAP_SHARED,0,0);
    if (buffer == NULL){
        printf("mmap error\n");
        return 1;

    }

    pid = fork();

    if(pid == -1){
        printf("fork error\n");
        return 2;

    }

    if(pid == 0){
        printf("give me a string:\n");
        gets(buffer);
        return 0;
    }

    wait(NULL);

    printf("%s\n",buffer);

    return 0;


}
```

L'applicazione chiama MMAP, vuole un certo numero di pagine, modalità lettura/scrittura, pagine ANONIME (noi vogliamo che compaia nulla all'interno di queste pagine, quindi contenuto inizialmente unico), ma la mappatura è SHARED. Il che implica dire che poi quando facciamo la fork(), il processo figlio chiederà di dare una stringa, la stringa viene letta e scaricata nella variabile BUFFER di cui abbiamo il pointer, ossia la zona di cui abbiamo appena m-mappato.
Poi usciamo. Il parent fa una wait ed aspetta che il child abbia completatp, e poi manda in printf() una stringa, ossia la stringa contenuta all'indirizzo di "BUFFER".
Attenzione, qui sono due processi. Uno dei due è mappato nella memoria ha generato con fork un altro processo, ma questa memoria è condivisa. Quindi c'è una reale istanza singola di questa informazione di memoria. Quindi il processo child che accede utilizzando buffer, accede alle sue pagine logiche ma l'istanza è unica in termini di pagine fisiche. E il parent accede alle sue pagine logiche ma anche lui ha l'unica istanza di queste pagine che sono presenti nella memoria fisica.
La stringa viene acquisita dal child e rimandata in printf dal parent. Perché il child le ha scritte su delle pagine che sono "shared" non "copy_on_write".

```
AGEMENT/UNIX/mmap> ./a.out
give me a string:
xxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxx
```

Se togliamo MAP_SHARED il processo child nella gets() va a scrivere sulle sue pagine logiche che corrisponderanno ad una copia privata di informazioni che non sarà vista dal parent quando andremo in printf().