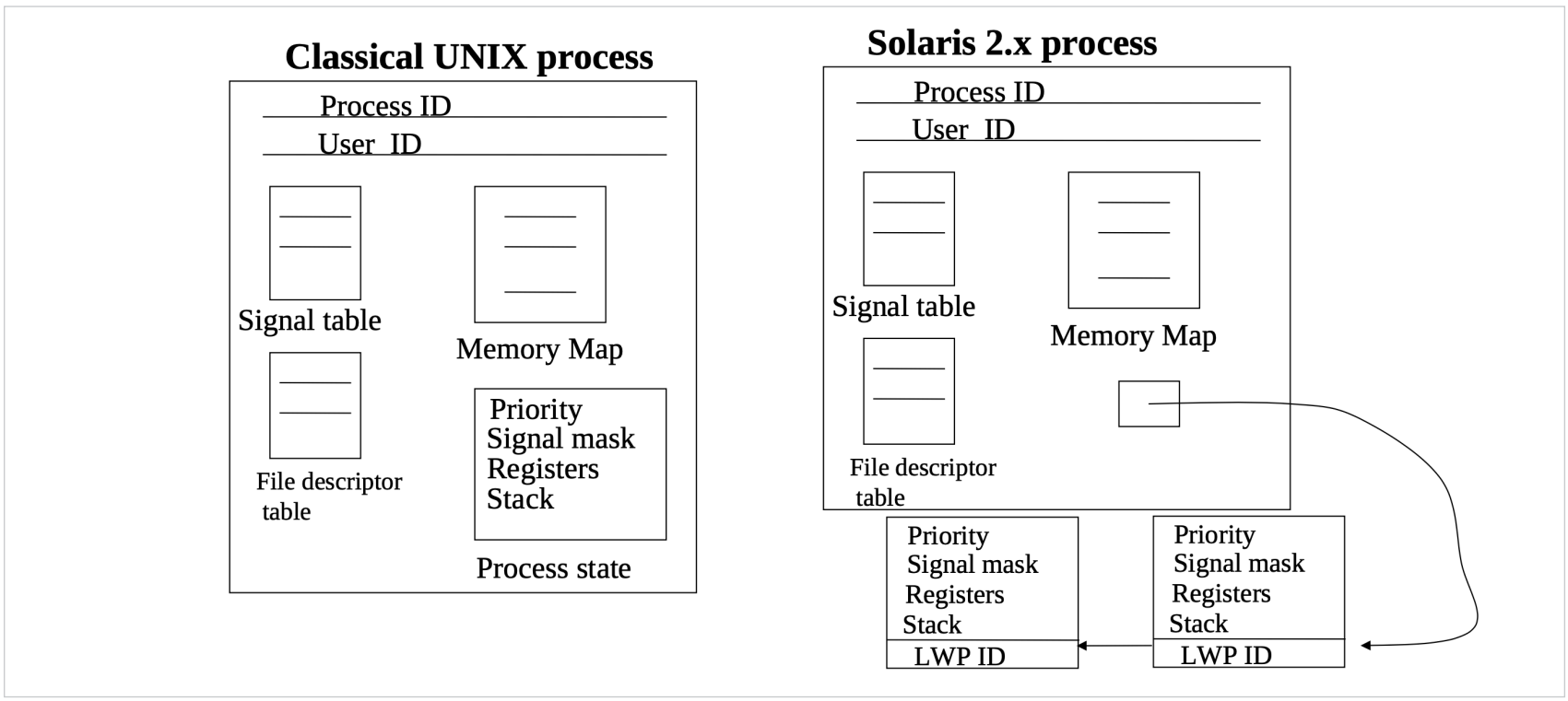
ESEMPIO

sabato 8 aprile 2023 10:26



Per generare un thread all'interno di un sistema operativo UNIX noi abbiamo un processo che è già attivo, eventualmente abbiamo un thread che è già attivo, l'unica cosa che dobbiamo fare a livello sistema è quella di avere la possibilità di risalire ad un'area di memoria utilizzabile per mantenere il CONTESTO del nuovo percorso d'esecuzione: dobbiamo avere anche una stack area disponibile a livello sistema.

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <unistd.h>
#include <string.h>
#include <sys/wait.h>

#define SPAWNS 50000

void* child_thread(void*p){
    return NULL;
}

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

   int i;
   int status;
   void *thread_status;
   pthread_t tid;

   if (argv[1]==NULL){
    printf("missing arg[1]\n");
    exit(EXIT_FAILURE);
   }

   if (strcmp("processes", argv[1]) ==0) {
    for (i=0;i<SPAWNS;i++){
          if(fork()) wait(&status);
      else exit(0);
    }
   }

   if (strcmp("threads", argv[1]) ==0) {
    for (i=0;i<SPAWNS;i++){
      pthread_create(&tid,NULL,child_thread,NULL);
      pthread_join(tid,&thread_status);
    }

   }
}
```

Il thread vive nella funzione child_thread e in questa funzione non facciamo nulla, se non ritornare.

Abbiamo la necessità di passare un ARGV[1], andiamo ad effettuare una STRCMP e vediamo se in ARGV[1] è stata passata la stringa “processes” e se è così stiamo lavorando per andare a verificare quant’è la latenza che impieghiamo per lanciare processi.
Quindi all’interno del ciclo FOR, if(fork()) wait. Altrimenti usciamo.

Questo è un modo per avere un parent che lancia un figlio e poi fa La wait.
Stiamo facendo la EXIT all’interno del figlio e la WAIT all’interno del padre.
Se questa cosa noi la facciamo per tanto tempo all’interno di un for possiamo misurare il tempo che il sistema può aver speso per eseguire queste attività.
Se argv[1] è threads, quindi non è processes, facciamo un for come prima, ma ora all’interno creiamo un thread e lo aspettiamo.

```
THREADS/UNIX> gcc threads-performance.c -lpthread
```

Come facciamo a misurare da una shell il tempo d'esecuzione di un'applicazione?

```
THREADS/UNIX> time ./a.out processes
```

ARGV[0]="./a.out"
ARGV[1]="processes"

```
OCESSES-AND-THREADS/UNIX> time ./a.out processes
```

```
real    0m16.624s
user    0m5.408s
sys     0m2.609s
```

```
OCESSES-AND-THREADS/UNIX> time ./a.out threads

real    0m1.855s
user    0m0.102s
sys     0m0.429s
```

Con threads la quantità di tempo che l'applicazione impiega per eseguire tutte le sue attività per spwnare 5000 threads, è inferiore rispetto allo spawn di 5000 processi. Ma è anche ovvio.