

THREAD) \rightarrow PIÙ PROCESSI

PROCESSI \rightarrow MOLTIPIGLI FLUSSI
DI ESECUZIONE

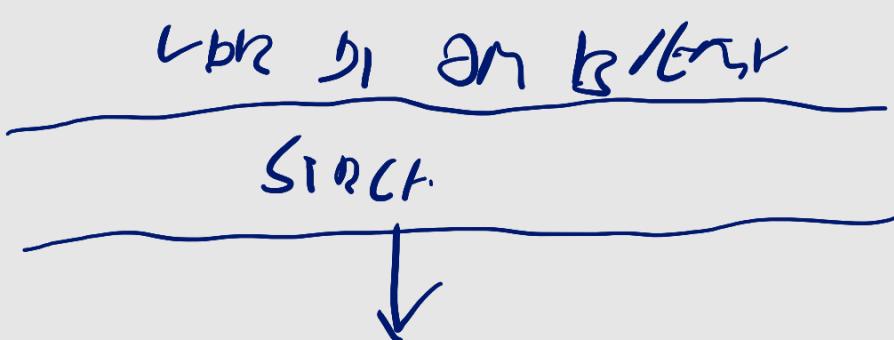
↳ INSERIRE UN PROCESSO
CHE CONDIVIDE IL MESES

\rightarrow ESECUZIONE DELLO STESSO

PROGRAMMA \rightarrow SIMULAZIONE DI:

HIGH LEVEL

COMPILAT.

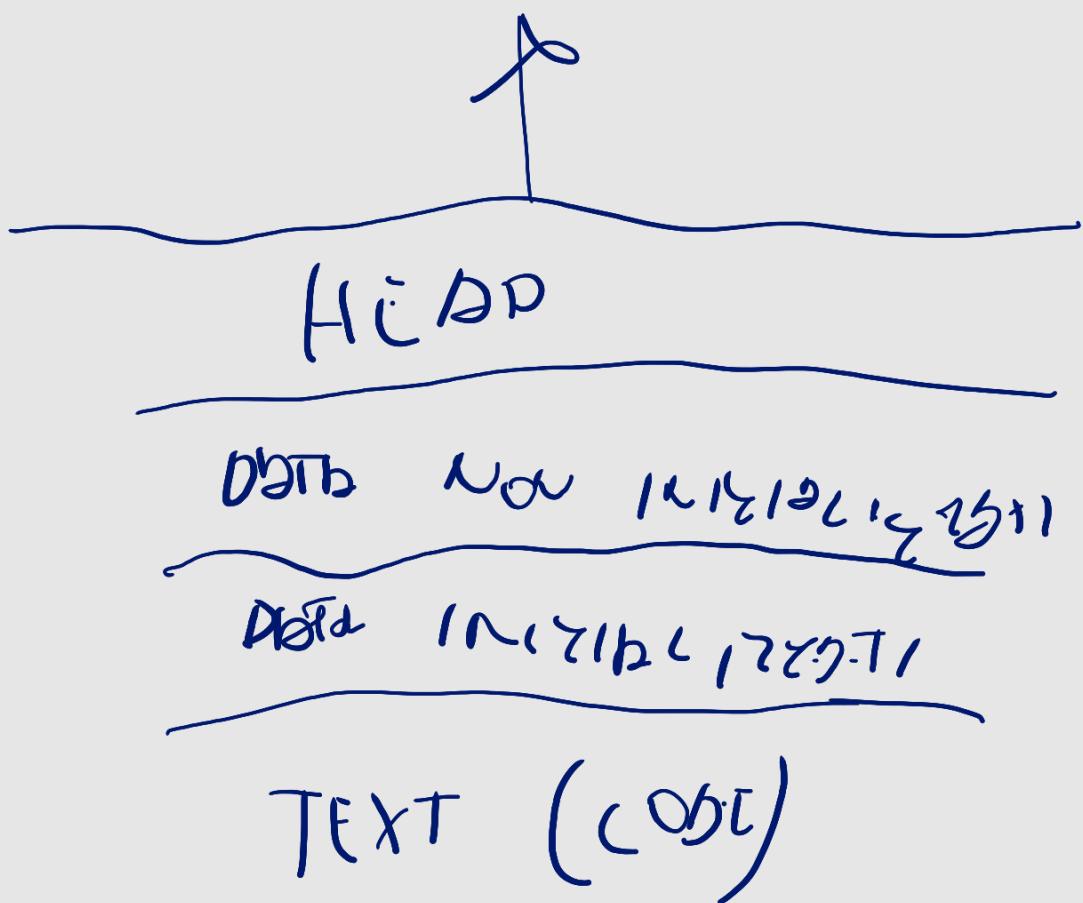


STACK TH3

STACK TH2

STACK TH1

LIBRI DI AMBIENTI CONDIVISI



STRUCTURES

→ STACK SINGLE

→ OPERATE SU REGISTRE!
E KR UN PROGRAMM COUNTER

			<u>CAP.</u>	<u>DATA</u>	<u>FILES</u>
CODE	DATA	FILES	Dir	Reg	Rec
DATA	PROG CONT.	STACK	Stack	Stack	Stack
P			PC	PC	BC

Th sema

→ COMUNICACION MÁS FÁCIL
DEI PROCESSI

{
 | MUTEX
 | CONDICION VARIABLE
 | SEMAFORO

VAR GLOBALE

KERNEL / USER THREAD

L'USER DÀ TUTTO AL TRUTH

LIBRARY E IL KERNEL

PER ESSERE SCURO

LINUX PTHREADS → (KERNEL THREADS)

↪ UN PROCESSO PUÒ SVOLGERE
PIÙ TIESTE IN PARALELO

Concurrenz

↳ OR THE objects in rot

o length 2 MIND

→ MULTI-CORE - 

EXECUTION PARALLEL

CONDIVISION THREADS

(MESSO PROCESSO)

→ AUTOMORPHIC DESIGN

→ PWD < PWD → PRIVILEGI

→ FILE → WORKING
DIRECTIONS

THREAD → ID

(link with obj memory)

CREAZIONE THREAD

(PTHREAD_T *THRD), PTHREAD_ATTR_T)

PUNTO A → THRD ID → ATTR
PUNTO B → PUNTO C → ATTRIBUT
PUNTO D → FUNZ → POSITION

VARI * (*_{CHAR} / VARI *)

↳ RETURN

VOID * PTR

↳ PUNT A UN TIPO DI DATO
E PLICATI

PTHREAD_CREATE() → Funzione
libreria

PTHREAD_CREATE()

↳ Ogni thread ha
proprio logica

CREA UN NUOVO

PROCESSO ETTO

CONTROLLARE LA MEMORIA

CHE IL FERMI

THREAD

TERMINAZIONE THREADS

↳ EXIT() / CANCEL()

↳ PTHREAD_EXIT(VOID * TH)

PROPRIO ID

↳ PTHREAD_SETID()

PITREBBEO JOLT (Threads, void *RET)

↳ attend alle termini

VAL DI RETORNO SE

Processo ha wait nel

PROCESSO

→ Quando THREAD può fare un
JOLT su un altro

Sistemi Operativi - Martino Trevisan - Università di Trieste

Esempi

Creazione di un thread

```
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

static void * threadFunc(void *arg){
    printf("From Thread: %s", (char *) arg);
    int * ret = malloc(sizeof(int));
    *ret = strlen(arg);
    return ret; // Valore di ritorno del thread
    // Equivale a pthread_exit(ret);
}

int main(int argc, char *argv[]){
    pthread_t t1;
    void *res; // Per valore di ritorno
    int s;

    s = pthread_create(&t1, NULL, threadFunc, "Hello world\n"); // Creazione
    if (s != 0){
        printf("Cannot create thread");
        exit(1);
    }

    printf("Message from main()\n");
    s = pthread_join(t1, &res); // Join. Richiede un void **, ovvero &res
    if (s != 0){
        printf("Cannot join thread");
        exit(1);
    }
    printf("Thread returned %d\n", *((int *)res)); // Utilizzo del valore di ritorno
    free(res); // Needed as that zone was allocated with malloc
    exit(0);
}
```