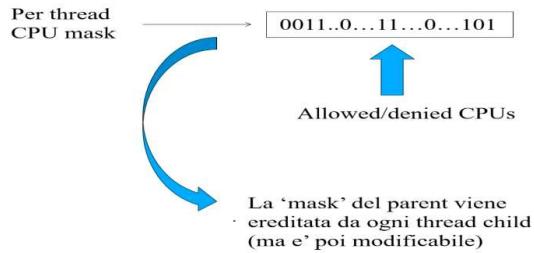


Scheduling 6 Affinità processore

martedì 28 ottobre 2025 15:12

Andiamo a vedere ora dell'**affinità del processore** ovvero la situazione nella quale un thread è affine (possibile esecuzione su quella CPU) ad una determinata CPU , venendo messo nella coda opportuna della CPU , la quale scelta viene fatta sulle politiche interne del SO. Quindi si può simulare uno scenario dove abbiamo meno CPU core rispetto a quelle reali . Quindi sfruttando questa assegnazione ad un sotto insieme delle CPU core , si ha la possibilità' di riservarne alcune per svolgere azioni critiche. Ma come viene decisa questa affinità tra thread e CPU ? Attraverso una maschera di bit , la quale descrive su quali CPU core il thread può girare (bit a 1 si , bit a 0 no):



Vediamo come :

1. UNIX

SCHED_SETAFFINITY(2) Linux Programmer's Manual SCHED_SETAFFINITY(2)
NAME top
sched_setaffinity, sched_getaffinity - set and get a thread's CPU affinity mask
SYNOPSIS top
#define _GNU_SOURCE /* See feature_test_macros(7) */
#include <sched.h>
a. int sched_setaffinity(pid_t pid, size_t cpusetsize,
const cpu_set_t *mask);
int sched_getaffinity(pid_t pid, size_t cpusetsize,
cpu_set_t *mask);
DESCRIPTION top
These are the backend of the taskset shell command

2. WINDOWS

Syntax
C++
BOOL WINAPI SetProcessAffinityMask(
 In HANDLE hProcess,
 In DWORD_PTR dwProcessAffinityMask
);
a. Parameters
hProcess [in]
A handle to the process whose affinity mask is to be set. This handle must have the PROCESS_SET_INFORMATION access right. For more information, see Process Security and Access Rights.
dwProcessAffinityMask [in]
The affinity mask for the threads of the process.
On a system with more than 64 processors, the affinity mask must specify processors in a single processor group.

b. Analogamente per i thread:

Syntax
C++
DWORD_PTR WINAPI SetThreadAffinityMask(
 In HANDLE hThread,
 In DWORD_PTR dwThreadAffinityMask
);
i. Within current processor group

c. Vediamo esempio : process priority (cambio entrambe le priorità)

```

// processes-priorities.cpp : Defines the entry point for the console application.

#include <windows.h>
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    BOOL newprocess;
    STARTUPINFO si;
    PROCESS_INFORMATION pi;
    int i = 0;
    char p = (char) argc;
    char pwd[MAX_PATH];
    char cpul[128];
    int ret;
    GetCurrentDirectory(MAX_PATH, pwd); printf("process is %d - current directory is: %s\n", GetCurrentProcessId(), pwd);
    /* print argv elements */
    while (i < argc){
        printf("arg %d is: %s\n", i, argv[i]);
        i++;
    }
    if (argc>1){
        memset(&si, 0, sizeof(si));
        memset(&pi, 0, sizeof(pi));
        si.cb = sizeof(si);
        printf("trying running the child command: %s\n", argv[1]);
        fflush(stdout);
        newprocess = CreateProcess(argv[1], "child", NULL, NULL, FALSE, NORMAL_PRIORITY_CLASS, NULL, NULL, &si, &pi);
        if (newprocess == FALSE){
            printf("CreateProcess failed - no child command has been run\n");
            ExitProcess(1);
        }
        printf("created new process\n");
        while (1){
            printf("please give me the new priority to set for child process\n");
            fflush(stdout);
            scanf("%s", prio);
            if (strcmp(prio, "none") == 0) break;
            if (strcmp(prio, "high") == 0){
                ret = SetPriorityClass(pi.hProcess, HIGH_PRIORITY_CLASS);
            }
            else{
                ret = SetPriorityClass(pi.hProcess, NORMAL_PRIORITY_CLASS);
            }
            printf("set priority returned %d\n", ret);
        }
        TerminateProcess(pi.hProcess, 0);
        WaitForSingleObject(pi.hProcess, INFINITE);
        ExitProcess(0);
    }
    else{
        while (1);
    }
}

```

- ii. Il quale una volta creato il processo, chiede all'utente di assegnarli una priorità, la quale poi verrà vista nel task manager

d. Processor Affinity :

```

// processor-affinity.cpp : Defines the entry point for the console application.

#include <windows.h>
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    BOOL newprocess;
    STARTUPINFO si;
    PROCESS_INFORMATION pi;
    int i = 0;
    char p = (char) argc;
    char pwd[MAX_PATH];
    char cpul[128];
    int mask;
    unsigned long long mask;
    GetCurrentDirectory(MAX_PATH, pwd);
    /* print argv elements */
    while (i < argc){
        printf("arg %d is: %s\n", i, argv[i]);
        i++;
    }
    if (argc>1){
        memset(&si, 0, sizeof(si));
        memset(&pi, 0, sizeof(pi));
        si.cb = sizeof(si);
        printf("trying running the child command: %s\n", argv[1]);
        fflush(stdout);
        newprocess = CreateProcess(argv[1], "child", NULL, NULL, FALSE, NORMAL_PRIORITY_CLASS, NULL, NULL, &si, &pi);
        if (newprocess == FALSE){
            printf("CreateProcess failed - no child command has been run\n");
            ExitProcess(1);
        }
        printf("created new process\n");
        while (1){
            printf("please give me the new CPU id to set\n");
            fflush(stdout);
            scanf("%s", cpul);
            if (strcmp(cpul, "none") == 0) break;
            if (strcmp(cpul, "notone") == 0){
                mask = 0x0;
            }
            else{
                mask = 0x1;
            }
            ret = SetProcessAffinityMask(pi.hProcess, mask);
            printf("set affinity in mask %p returned %d\n", (void*)mask, ret);
        }
        TerminateProcess(pi.hProcess, 0);
        WaitForSingleObject(pi.hProcess, INFINITE);
        ExitProcess(0);
    }
    else{
        while (1);
    }
}

```

- ii. Che una volta lanciato , viene chiesto all'utente se vuole cambiare CPU ed in base a quella che digita il sistema la cambia; per verificarla si va nel task manager