第一讲 绪论

2023年2月19日 10:00

```c
#include <stdio.h>
#include <stdlib.h>
#include "common.h"

int main(int argc, char *argv[])
{
    if (argc != 2)
    {
        fprintf(stderr, "usage: cpu <string>\n");
        exit(1);
    }
    char *str = argv[1];

    while (1)
    {
        printf("%s\n", str);
        Spin(1);
    }
    return 0;
}
```

```c
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include "common.h"

int main(int argc, char *argv[])
{
    if (argc != 2)
    {
        fprintf(stderr, "usage: mem <value>\n");
        exit(1);
    }
    int *p = (int*) malloc(sizeof(int));
    assert(p != NULL);
    printf("(%d) addr pointed to by p: %p\n", (int)getpid(), p);
    *p = atoi(argv[1]); // assign value to addr stored in p
    while (1)
    {
        Spin(1);
        *p = *p + 1;
        printf("(%d) value of p: %d\n", getpid(), *p);
    }
    return 0;
}
```

```c
#include <stdio.h>
#include <stdlib.h>
#include "common.h"
#include "common_threads.h"

pthread_mutex_t mymutex = PTHREAD_MUTEX_INITIALIZER;

//volatile int counter = 0;
int counter = 0;
int loops;

void *worker(void *arg) {
    int i;
    for (i = 0; i < loops; i++) {
        counter++;
    }
    return NULL;
}

int main(int argc, char *argv[]) {
    if (argc != 2) {
     fprintf(stderr, "usage: threads <loops>\n");
     exit(1);
    }
    loops = atoi(argv[1]);
    pthread_t p1, p2;
    printf("Initial value : %d\n", counter);
    Pthread_create(&p1, NULL, worker, NULL);
    Pthread_create(&p2, NULL, worker, NULL);
    Pthread_join(p1, NULL);
    Pthread_join(p2, NULL);
    printf("Final value   : %d\n", counter);
    return 0;
}
```

```
zxs@localhost chap1]$ ./th
Initial value : 0
Final value   : 2000
[zxs@localhost chap1]$ ./t
Initial value : 0
Final value   : 2000
[zxs@localhost chap1]$ ./t
Initial value : 0
Final value   : 2000
[zxs@localhost chap1]$ ./t
Initial value : 0
Final value   : 161394
[zxs@localhost chap1]$ ./t
Initial value : 0
Final value   : 145232
[zxs@localhost chap1]$
```

why?

- 软件设计模式
- 深入理解语言标准
- 算法

思想迁移到软、硬件平台各个领域

- web server
- Database
- cloudcomputing
- ……

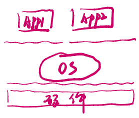
- 手持智能设备(手机、平板)
- 工业设备
- 医疗设备
- ……

OS
- → API
- → 进程、线程
- → 内存管理
- → 存储
- → 输入/输出
- → 隔离

web server
- → 接口
- → 多用户、多任务
- → 搜索缓存结构、同步
- → 网页、配置.
- → 网络
- → 客户端脚本 (sand box)

What?

位置：



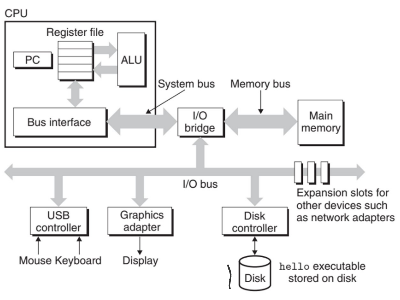
功能：

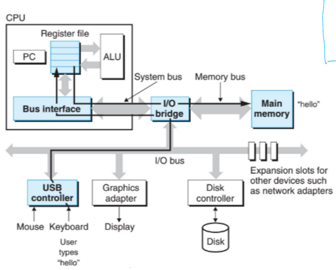
资源管理
- 硬件
  - 仲裁(分配)
  - 隐藏硬件限制(虚拟化)
  - 通用服务(标准接口)
- 软件
  - 调度
  - 隔离错误
  - 通信.

目的：资源高效利用

运行特征：最早启动. 特权. 最晚退出.

硬件与程序






```c
#include <stdio.h>

int main()
{
    printf("hello, world\n");
}
```

读取命令

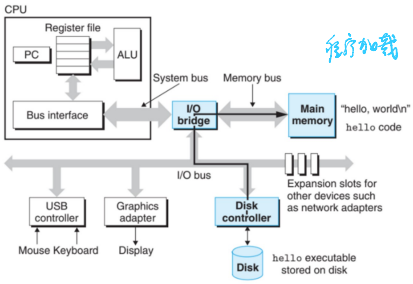
reads 1000

hreads 1000

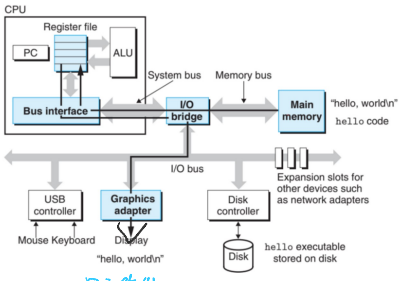
hreads 1000

hreads 100000

hreads 100000



程序加载



显示器

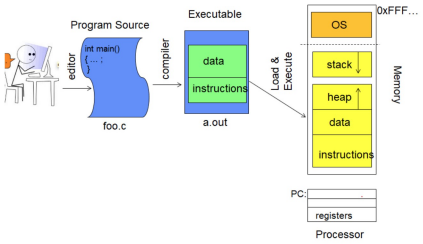
程序执行：(要素)

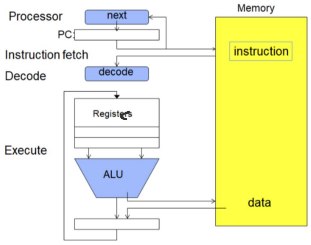
内存加载

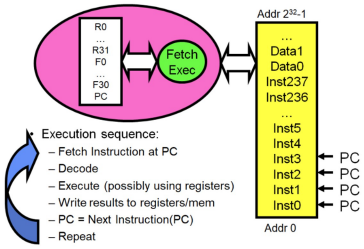
代码.

数据

栈

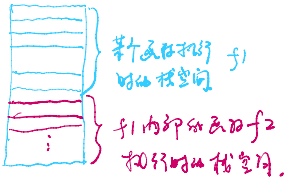






堆：为保存的内存区域，动态内存需求.

栈：为保存的内存区域，保存参数、局部变量空间分配. 栈帧：返回地址.



第n次执行f1时的栈空间

f1内部分别对f2执行时的栈空间.

```
1 void swap(int * a, int *b)
2 {
3         int c;
4         c = *a; *a = *b; *b = c;
5 }
6
7 int main()
8 {
9         int a, b;
10        a = 16; b = 32;
11        swap(&a, &b);
12        return (a - b);
13 }
```