1_0
A Tour of Computer Systems

- A computer system consists of hardware and systems software that work together to run application programs.
- This book is written for programmers who want to get better at their craft by understanding how these components work and how they affect the correctness and performance of their programs. You will be on your way to becoming a rare “power programmer".
- You are going to learn practical skills such as:
  - how to avoid strange numerical errors caused by the way that computers represent numbers.

```
maximelionel@Maximes-MBP ~ % lldb
(lldb) print 40000*40000
(int) $0 = 1600000000
(lldb) print 50000*50000
(int) $1 = -1794967296
(lldb) print 45000*45000
(int) $2 = 2025000000
(lldb) print 48000*48000
(int) $3 = -1990967296  // overflow - 32 bit system have a maximum
```

```
(lldb) print 300*400*500
(int) $4 = 60000000
(lldb) print 300*500*400*600
(int) $5 = 1640261632
```

  - how to optimize your C code by using clever tricks that exploit the designs of modern processors and memory systems.
    - memory must be allocated and managed, while many applications are memory dominated
    - Cache and virtual memory effects can greatly affect program performance

```c
#include <stdio.h>
#include <time.h>
#include <stdlib.h>

int src[2048][2048]; // allocate in heap
int des[2048][2048];

void copyij(int src[2048][2048], int det[2048][2048])
{
        int i, j;
        for(i = 0; i < 2048; i++)
                for(j = 0; j < 2048; j++)
                        det[i][j] = src[i][j];
}


void copyji(int src[2048][2048], int det[2048][2048])
{
        int i, j;
        for(j = 0; j < 2048; j++)
                for(i = 0; i < 2048; i++)
                        det[i][j] = src[i][j];
}

int main(void)
{
```

```
27          clock_t start_ij, start_ji, finish_ij, finish_ji;
28
29          start_ij = clock();
30          copyij(src, des);
31          finish_ij = clock();
32          printf("The duration for copyij is: %f seconds\n",(double)(finish_ij -
    start_ij)/CLOCKS_PER_SEC);
33
34          start_ji = clock();
35          copyji(src, des);
36          finish_ji = clock();
37          printf("The duration for copyji is: %f seconds\n",(double)(finish_ji -
    start_ji)/CLOCKS_PER_SEC);
38
39          return 0;
40  }
```

- adapting program to memory system can greatly lead major speed improvements
- memory referencing bugs

```
#include <stdio.h>

typedef struct
{
        int a[2];
        double d;
}struct_t;

double fun(int i)
{
        volatile struct_t s;
        s.d = 3.14;
        s.a[i] = 1099999989;
        return s.d;
}

int main()
{
        printf("f(0): %f\n", fun(0));
        printf("f(1): %f\n", fun(1));
        printf("f(2): %f\n", fun(2));
        printf("f(3): %f\n", fun(3));
        return 0;
}
```

```
parallels@ubuntu-linux-22-04-desktop:~/csapp/mem_ref_error$ ./a.out
f(0): 3.140000
f(1): 3.140000
f(2): 3.140000
f(3): 69909844.480000
```

- Reason - out of bound array index

- how the compiler implements procedure calls and how to use this knowledge to avoid the security holes from buffer overflow vulnerabilities that plague network and Internet software.
- how to recognize and avoid the nasty errors during linking that confound the average programmer.
- how to write your own Unix shell (命令行), your own dynamic storage allocation package, and even your own Web server.
- the promises and pitfalls of concurrency (并发), a topic of increasing importance as multiple processor cores are integrated onto single chips.

- help you understand what happens and why when you run hello on your system. - create hello.c in ubuntu and run

```
1   #include <stdio.h>
2
3   int main() {
4           printf("hello, world\n");
5           return 0;
6   }
```