Quick look up

Hexadecimal: Aa-10 Bb-11 Cc-12 Dd-13 Ee-14 Ff-15
4 bits nibble, 8bits Byte, 2 Bytes Word, 4 Bytes - Double Word
KiB MiB GiB TiB PiB $2^{10n} Bytes = 1024^n Bytes$
KB MB GB TB PB $1000^n Bytes$

Linux command

ltrace: lists the library calls made by a running process as it is running. **strace**: list the system calls.
touch: change timestamp / create an empty file.
cat, **sort**, **uniq**, **echo**
stat: information of a file. **file**: brief information of a file.
od: show content of file in various ways. -c character -d
dd: copy part of file. dd if=.. bs=BUFFER count=NBYTES of=..
cp: copy a whole file. **head**: filter first few lines of content.

Special Files

/dev/null: black hole. **/dev/zero**: all zeros.
/dev/random: full of random number.
/dev/mem: overview of main memory.
/dev/hda: raw block of hard disk.

rPeAnut

```
xvar : block #1 ; a global variable xvar initialized to 1
xvar: block 10 ; an array xvar of length 10, all values 0
str: block #''abcde'' ; apply for a string
load #xvar R0 ; load address of xvar to R0 (immediate load)
load xvar R0 ; load the value in xvar to R0 (absolute load)
store R3 xvar ; value in address R3 to xvar (indirect rest)
store R3 #xvar R0; global variable as base
store R0 #-1 SP ; store for return value in recursion
p1.x = 4 differ to pp1 -> x = 4 in what you load from
p1.x load from #p1 , p1 -> x load from pp1
```

negate:flip all bits then plus 1. **not**: just filp all bits
rotate:most significant bit will turn back to the least bit
status 0xFFF1, IO control 0xFFF2 represents?

File operation

```
/* scanf, fscanf, malloc, free */
scanf("%10s", p);
fscanf(fp,"%[^,]%*%lf\n", people[i].n, &people[i].h) != EOF
int* p = (int *) malloc ( sizeof(int) * 100 );
free(p);
/* open, read, write */
int open(const char * pathname, int flags, mode_t mode)
// returns a 'file descriptor', a small non-negative
//   number which provides a reference to the opened file.
int  read( int handle, void *buffer, int nbyte );
// returns 0 at the end of file
int  write( int handle, void *buffer, int nbyte  );
// The function returns the number of bytes written to file.
// A return value of -1 indicates an error.
// lseek: move current file offset
off_t lseek(int fd, off_t offset, int whence);
-------------------------------------------------------
/* File open example */
#include <stdio.h>
int main() {
    FILE *pFileS = fopen( "s.rar", "rb" );
    if(!pFileS ) return 1;
    FILE *pFileD = fopen( "d.rar", "wb" );
    unsigned char buffer[ 4 * 1024 ];
    int nRead;
    while(nRead = fread( buffer, sizeof(unsigned char),
            sizeof(buffer), pFileS) )
        fwrite(buffer, sizeof(unsigned char), nRead, pFileD);
    fclose( pFileS ); fclose( pFileD );
    return 0;
}
```

Network operation

inet_ntop() .
A string containing an IP address in readable form, and you want to pack it into a struct sockaddr_in or a struct sockaddr_in6. In that case, the opposite funcion inet_pton()
inet_ntop() returns the dst parameter on success, or NULL on failure (and errno is set).
inet_pton()returns1on success. It returns-1if there was an error (errno is set), or 0 if the input isn't a valid IP address.

```
// inet_add return a long integer
ina.sin_addr.s_addr = inet_addr("132.241.5.10");
/* reverse conversion by inet_ntoa */
a1 = inet_ntoa(ina1.sin_addr); // result is 198.92.129.1
printf("address 1: %s ",a1); // address 1: 132.241.5.10
-------------------------------------------------------
// IPv4 demo of inet_ntop() and inet_pton()
struct sockaddr_in sa;
char str[INET_ADDRSTRLEN];
// store this IP address in sa:
inet_pton(AF_INET, "192.0.2.33", &(sa.sin_addr));
// now get it back and print it
inet_ntop(AF_INET, &(sa.sin_addr), str, INET_ADDRSTRLEN);
printf("%s\n", str); // prints "192.0.2.33"
-------------------------------------------------------
// IPv4 demo of inet_ntop() and inet_pton()
struct sockaddr_in sa;
char str[INET_ADDRSTRLEN];
// store this IP address in sa:
inet_pton(AF_INET, "192.0.2.33", &(sa.sin_addr));
// now get it back and print it
inet_ntop(AF_INET, &(sa.sin_addr), str, INET_ADDRSTRLEN);
printf("%s\n", str); // prints "192.0.2.33"
```

Basics

FLOP: floating point operation per second.
Mbps: Mega bit per second. **MIPS**: Mega instruction persecond.
Endianness: the order the bytes within a single word are stored within main memory. **Big endianness** – most significant byte first.
Way to represent negative number:
1. simple offset: normal addtion does not work.
2. reserve 1-bit as sign bit: range -127 to 127.
3. 2s complement: take advantage of 1 and 2.

2s complement: range -127 to 128
negate value: flip all bits and plus 1.
overflow flag: result is too large to be represented.
Fixed point: $(-1)^s m b^e$
IEEE 754 floating point: 1 sign bit, 8 exp bits, 23 significand
subnormal: $(-1)^s \times 0.m \times 2^{e-126}$. normal: $(-1)^s \times 1.m \times 2^{e-127}$.
Special case: exp $0xFF$, if $m = 0$, value = inf. else, value = NaN

Introduction

Abstraction: with multiple levels of detail / description, it helps manage complexity and provides interfaces and standards.
Virtualization: gives the appearance of a capability or service. This decouples the service from the underlying physical resources. It helps in terms of simplicity, flexibility and resource sharing.
Prototypes: signatures of functions will be used in code.
Static keyword reduces the scope to the file it is within.

Files and File System

system call: is how a program requests a service from an operating system's kernel.
In Linuxe Based x86 machines, system calls are performed by generating interruption, which is trapped by kernel and then processed.
A **FILE** is a named collection of related information.
A **file system** provides a uniform logical view of related information.
Two ways to READ/WRITE files:
system calls: open, read, write, ioctl.
file streams: fopen, fread, fwrite.
mmap:file can be mapped into memory, done by the 'mmap' function. **mmap** enables the file's contents to be view and modified in normal memory.
SHARED mmap: modifications to the memory of the mapped file are written back to the actual file, also other processes that maps the same file sees the same modifications.
PRIVATE mmap: process has its own private copy of the file. Modifications are not written back to the file and other processes do not see any changes.
Fixed File Descriptor (int fd): stdin - 0,stdout - 1, stderr - 2
res = read(0,buf,255); res = write(1,"Hello",6);
3 access files methods:sequential/direct/memory mapped access

Structure of C program

Source → (COMPILE) → assemble code(.s) → (ASSEMBLE) → object code(.o) → (LINK) → executable
malloc returns a pointer to the amount of memory requested.(free)
struct may add padding to make certain types are appropriately aligned in memory.

Interrupts

Interrupt, *an event alters the normal fetch-decode-execution.* When occurred, current program will be suspended. caused by hardware event and software event.
Interrupt handler should be carefully designed such that the state of CPU returns exactly how it was found.
Interrupt latency: time between an interrupt occur and code executed to handle it.
Interrupt storm: Interrupts occur more frequently than the handler can serve, then requests to handle interrupts will be lost.

Memory

Dynamic random access memory (DRAM) only requires *a single transistor and capacitor per bit*, this enables much higher density memory (compared with SRAM). However, it is considerably slower and more complex to access.
Static random access memory (SRAM) is fast volatile memory that is simple to interface with. SRAM will be found in: CPU caches, buffers for various devices within a computer system.
ROM is non-volatile so state is maintained without power.
Flash memory is an electronic non-volatile computer storage device that can be electrically erased and reprogrammed.
Registers, L1 cache, L2 cache, DRAM, Flash Drive, Magnetic Disk Drive, Network Backup Server. (*capacity increase, speed decrease.*)

2 basic approaches for the design of the control unit.
Hardwire: these control units basically are finite state machines that are specially designed to sequence the CPU based on the instruction it is executing. (not particularly flexible and can be difficult to design, however, good performance)
Microcode: this approach uses ROM type memory within the CPU to act like machine code for sequencing the different action within the CPU. simpler and more flexible approach.

CPU performance

CPUs have moved caches onto the CPU die which enables the CPU

to be physically closer to the cache. This **reduces latency**.
Pipeling: involves in lots of independent stages, IF, ID, E, WB.
Superscale architectures involve duplicate functional units within the cpu and then start one instruction on the same clock cycle in the pipeline. Enables larger throughput of instructions.
Sometimes instructions will require data from memory before they can execute, this will stall the pipeline. **Slow the CPU**.
Outer of order execution approach loads the next few instructions and starts executing the instruction that has the required data.
Multithread: CPUs can maintain the programming context of multiple threads (so *duplication of state and register information*), without the duplication of processing units, caches, TLBs, etc, This enables multiple threads to be executed within the one core and hide latency. **Multicores**: The CPU can be duplicated, share the same memory and L2 Cache.

CPU Cache
Cache hit: occur when data required by instruction is in cache.
Cache miss: occur when data required is *not* in cache.
Write-through: written to cache and main memory at same time.
Write-back: data is just written to cache.

Linking
Binding of instructions and data to memory addresses may occur at: compile time, load time, or execution time.
Linking for libraries: **statically linked**: the library is compiled into the final binary executable. **dynamically linked**: just a 'stub' is includes in the binary executable, the library code is obtained as needed during execution.

Processes
A context switch changes the process a CPU is executing.
A **thread** is the basic unit of execution within a process.
Lightweight process (LWP, traditional, has only one thread).
Each thread, consists of a program counter, a register set and a stack space.
A **process** consists of a group of threads. They share text, data, opened files, capabilities, etc.
CPU-cache(SRAM) stores copies of main memory data..
Working set is the memory that a program is currently using fully associative cache, direct mapping cache, set associative cache.

Memory Management and Paging
Addresses generated by the CPU are called **logical addresses**. These are the addresses 'seen' by the user's programs.
Addresses seen by the main memory are called **physical addresses**.
Physical memory is divided into *fixed-sized blocks called frames.*
Logical memory is div into *blocks of the same size called pages.*
Logical addresses (produced by the CPU) are divided into *two parts: the page number and the offset.* **Each process has a page table**.
Memory Management Unit: responsible for mapping from logical address to physical address.
Page table: responsible for logical address to physical address. If it is large, page table could be stored in the main memory.
Page Table Base Register (PTBR): points to the page table. A fast lookup cache may store the page frame mappings within CPU.
Translation Look-aside Buffer (TLB): entry is the frame number.
Internal Fragmentation: the whole frame occupation by process causes waste.
Segmentation: A user or process would like to view memory as a set of variable-sized segments. Each segment has a name and there is no necessary ordering among segments.
Advantages of separating logical & physical addresses:
1.execution-time address binding. 4.simplifies sharing of data.
2.simplifies the swapping of processes in and out of memory.
3.protection / security of data between processes.

Virtual Memory
Swapping is an approach that involves bringing an entire process from disk into main memory so it may be executed.
Demand-paging is a lazy "swapper" which brings into memory the pages of the process that are needed.
One approach to paging is to only bring pages into memory when they are needed. This is called Pure demand paging.
Advantages of paging: 1. programs can be larger than physical memory
2. abstracts main memory into an extremely large logical storage area
3. increases the degree of multi-programming
swap space generally is faster than the file system (as file lookups and indirect allocation methods are not used)
page replacement methods:
1. frame-allocation algorithm
2. page-replacement algorithm(FIFO, OPT/MIN, LRU, LFU)
valid-invalid bit: hardware support

CPU Scheduling
In **multiprogramming systems**, the running task keeps running until it performs an operation that requires waiting for an external event (e.g. reading from a tape) or until the computer's scheduler forcibly swaps the running task out of the CPU. (multiprogramming is a CPU scheduling strategy.)
There are two **types of schedulers**:
1. non-preemptive - CPU is not 'forcefully' taken from the process.
2. preemptive - CPU may be forcefully taken from the process and switched to another process.
Criteria of scheduling algorithms:
1. CPU utilization - percentage of time the CPU is in use.
2. Throughput - rate at which processes are being completed.
3. Turnaround time - interval of time from the starting a process to completing the process.
4. Wait time - amount of time a process spends in the ready queue.
5. Response time - The amount of time it takes a process to start responding. This does not include the time to output the response.
Three algorithms for CPU Scheduling:
1. First Come First Serve (FCFS)
2. Shortest Job First (SJF) provably shortest waiting time
3. Round Robin (RR): each time CPU excution no more than a threshold.

Most **Ethernet controllers** have unique 48bit MAC address.
Ethernet's topology started as *bus* topology. But now LAN would generally use *star* topology with a switch at the center.
Weakness of bus topology: Only one device may transmit information to the bus at any time. If two units transmit at the same time a collision will occur, both transmission with usually be compromised and will need to be re-transmitted.

Particular IP Address:
0.0.0.0 - This host (only source), 127.X.X.X - loopback.
10.X.X.X - private network, 192.168.X.X - private network.
255.255.255.255 - Broadcast on the local network.
IPv4 (Internet Protocol) provides *32bit addresses (decimal)* for addressing computing devices that span the globe.
TCP (Transmission Control Protocol) provides a way of sending larger amounts of data to particular 'ports' on a host. and resend missing packets and check that packets arrive in the correct order.
UDP (User Datagram Protocol): unreliable way, without a connection to be setup, not order check but very fast.

1-bit register component diagram