objectives
① Learn how an OS manages resources.
② Experiment with buffer overflows.
③ Explore concurrency and asynchronous behavior
④ using OS terminology.
⑤

chapter01. Technology's Impact on Programs

Fundamental changes in commercial software

- 분산된 시스템으로 환경이 변경됨 → 더 작고 덜 비싼 기계들로
  Large DB, Business application
- Teminals with GUI and multimedia
- CSCW: computer supported cooperative work (협업작업 SW) ex. 공동편집, 공동작업
  - standalone applications with network communications
    - In conclusion, standalone applications are **a type of software program that is designed to run on a single computer or local machine of the user, without the need for a server or internet connection**.

Developments in tech. rely on communication concurrency and asynchronous operations

Terminology ①
- Asynchronous operation(비동기식 event)
  - : 예측불가한 event들이 발생 ↔ synchronous

    ⇒ interrupt가 작용하여 os가 좀 더 효율적으로 일하도록 함.

    ex) user key input, printing request ⇒ 사용자가 언제 누를지 모름

② Terminology
- Concurrency(동시성, 병행성) ↔ parallel task(병렬, 실제로 동시에 실행됨)
  - sharing of resources in the same time frame ⇒ 동시에 실행되는 것처럼 보임
  - peformance 성능이 높아짐

ex) cpu(여러 개의 process가 하나의 cpu를 공유하며 실행), data, codes, devices

③ • Communication

  ◦ conveying information by one entity to another

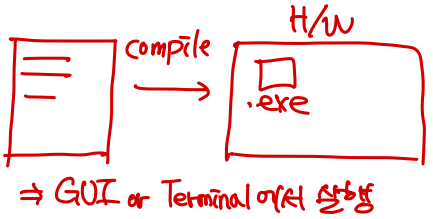
여러 개의 resource가 공유해며 실행 → performance 성능↑

Time and speed

- OS manages system resources (s/w(invisible) and h/w(visible))
- **past** : Disk drives
  - 자연적으로 한계가 있음. 기하급수적으로 access time이 감소할 수 없음
- **now** : processor
  - speeds are increaing exponentially (ex. cpu)

*CPU core가 하나라고 가정한 수업형

⇒ ready된 process가 여러 개



⇒ GUI or Terminal에서 실행

- **Multiprogramming** : cpu에 전달된 process가 여러 개

↔ Singleprogramming : cpu에 전달된 process가 단 하나

① ◦ Instruction cycle

  ▪ file → (complie) → .exe ⇒ GUI or Terminal에서 실행 → (ready queue에서 대기)

  → memory Load 후 process를 cpu에 실행

② ◦ OS는 ready process 중 하나를 choose → scheduling algorithm 사용

③ ◦ a resource request (read or write) results in an OS request (i.e a system call)

  ▪ (read or write) ⇒ I/O 작업 → h/w resource → os가 관리

  ◦ A system call is a request to the OS for service that causes the normal CPU cycle to be interrupted and control to be given to the OS

Time sharing

- 동시에 process 실행되는 것처럼 보이기 위해 엄청 빠른 속도로 process switch
- time quantum(적당한 값으로 주면 good)

Interrupts

asychronous 처리를 위해 존재 ⇒ 예측 불가한 event 처리를 위해 존재

- processor instruction cycle
  - the execution of a single instruction in a program
- A peripheral device(주변 기기, i/o device들을 말함) generates an interrupt tp set a hardware flag within the processor
- on each instruction cycle, the processor checks h/w flags.
- If interrupt occurs, the processor
  - saves current value of the program counter(저장소) and
  - loads the address of interrupt service routine.
  - cpu가 interrupt에 의해 잠깐 중지 → interrupt 수행 → 다시 process 수행
  - cpu가 중지되었을 때 진행 중 process save(즉, program counter에 저장)
- After finishing interrupt service routine, the processor resumes the execution of the previous instruction
  - 진행 중이었던 process 다시 실행

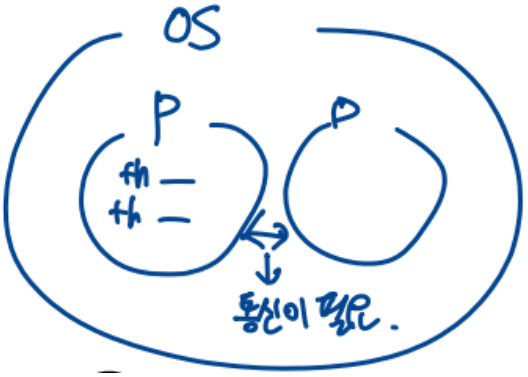
signals

- signal : software notification of an event
  - often a response of the OS to an interrupt.

ex) Ctrl+C(interrupt signal) → interrupt by the device dirver → OS sending a signal to the process

- synchronous : illegal instruction or a devide-by-zero ⇒ 강제로 끝냄
- asynchronous : Ctrl+C

Processes



- threads : 하나의 process는 적어도 하나의 thread, 즉 실행 흐름(=instruction의 순서)
- 프로그램의 특성에 따라 multi process or multi threads(concurrency)
- **Concurrent execution in UNIX**
  - to create multiple processes by fork()
    - process들은 대부분 부모 자식 관계의 tree 형태로 이루어져있음
    - 부모 자식 관계의 process들은 pipe 기법으로 통신
    - 부모 자식 관계가 아닌 process들은 IPC로 통신
      - IPC(interprocess communication) : signals. FIFOs, semaphores …
- multi threads(concurrency) → concurrency within a process
  - task 수행 효율이 올라감 ⇒ process 보다 안에서 여러 thread ⇒ task 수행효율 ↑

The network as the computer

- Concurrency and communication meet to form new application ⇒ network

- typical model is client/server model
  - ① a server process manages resources(자원들을 관리)
  - ② client processes access resources by sending a request to a server (서버에게 리퀘스트를 보냄으로써 자원들에 접근하는 고객 서버)
  - ③ the server performs the request and sends a reply to the client

⇒ network가 computer에서 하는 일

Fault tolerance(관용.)

ex) malloc 사용 후 free( ) 하는 것과 같은 예시

- 동적할당 메모리 해제와 같은 자원을 해제하는 것은 에러가 일어나기에 프로그램이 멈추지 않도록 에러 핸들링을 해야 함. ⇒ system이 고장나도 recovery
- C에서는 해당 사항을 check.
- C는 program을 변수의 범위에서 벗어나게 작성해도 되도록 함.

↔ Java : 변수 범위 check하는 runtime 존재

Buffer overflows

null 자리도 포함!

- 일어나지 않도록 input str에 format specification limits가 필요.
- consequences of buffer overflows
  - programs generally allocate automatic variables on the program stack
  - stack : 낮은 메모리부터 높은 메모리까지 모두 가능
    - the extra bytes may write over unused space, other variables, the return address or other memory 은 접근불가
  - 예기치 못 한 일 발생 가능
- security of buffer overflows ⇒ 왜?
  - 이를 적용하여 create a shell with root privileges.