Summary:-

Paging
Page Tables, Page Table Entries
TLB Cache
Swap Area (Backing store for pages)
Virtual Memory
Page Fault, Page Fault Exception Handler
Lazy Allocation/Demand Paging
Thrashing
Hierarchical Page Tables

Locality Principle
Major, Minor Faults
Copy on Write
Dirty Bit
Page Locking (Pre-Fetching)
Page Mapping -- Anonymous vs File
-- Private vs Shared Mapping

In x86

32 bit addr field

20 bit page no. +
12 bit offset
---
10 bit primary +
10 bit secondary +
12 bit offset

Page out
Page In

Swap out
Swap In

Linux Memory Zones:-

ZONE_DMA — Must be contiguous

ZONE_NORMAL (LOW Memory) — May not be contiguous
But can't be in swap area

ZONE_HIGH (HIGHMEM) — Not contiguous, can
be paged out (page faults)

e.g. 4 GB Memory
- Around 896MB ==> LOWMEM (Around 25%)
- Rest of mem ==> HIGHMEM (Around 75%)

Userspace memory allocation:-
mmap system call:-
- Anonymous + Private
- Anonymous + Shared
- File Mapping + Private
- File Mapping + Shared

File Mapping ==> File contents are mapped to virtual pages
==> Any read/write to virtual pages will reflect
to actual file on disk (post sync up for write)

Anonymous Mapping ==> Blank Virtual Pages

MAP_ANONYMOUS
MAP_PRIVATE
MAP_SHARED

Hands-on:-

example3a.c	==>	file + private
example3b.c	==>	anonymous + private
example2.c	==>	shared + anonymous, between parent & child
example1a.c	==>	anonymous + private, observe vsz, min faults
example1b.c	==>	file + private (Let's skip for now)

example3a - fix:-
    PROT_WRITE|PROT_READ
    file should have more than 26 chars
example3b - fix
    PROT_WRITE|PROT_READ

In case of shared memory, during fork, PTEs will be duplicated , but
    both PTEs will point to same frames (actual sharing)

Shared Memory b'n unrelated process (no parent-child)

shm_open
shm_close
---
shm_unlink

Analysis:-

shared memory
* userspace buffer
+ no system calls required, simple memory operations (faster)
- no synchronization support (race conds may occur, sequencing may not be enforced)

message queues & pipes
+ kernel space buffer
# send/write, recv/read are system calls
+ sync support is there , e.g. recv can block if Q is empty send can append

message queues vs pipes
MQ - descrete messages (no merging/splitting)
Pipes - continous stream of data (data merges onw write, split on read)