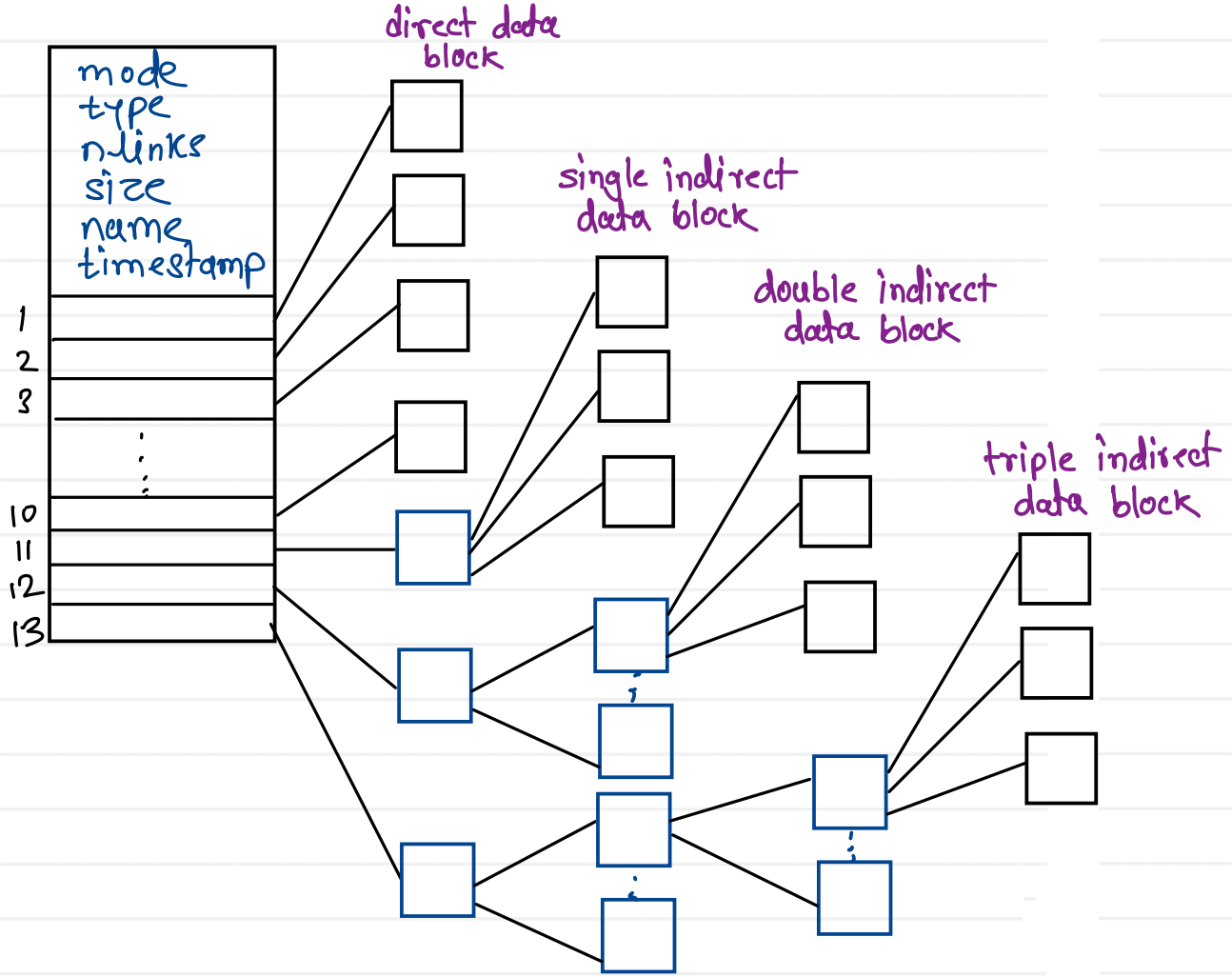
Sunbeam Institute of Information Technology
Pune and Karad

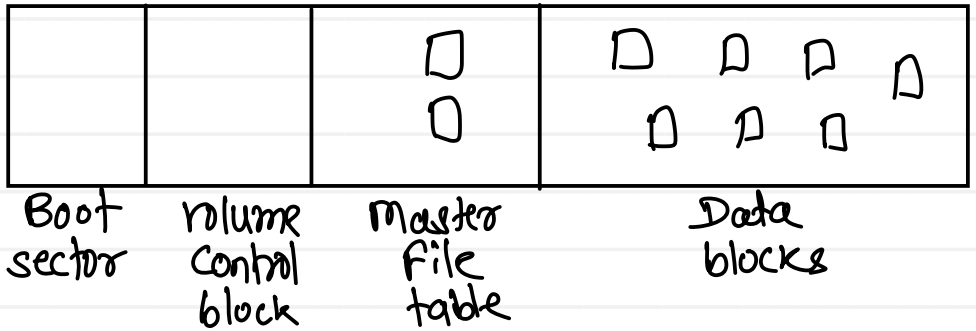
Module - Embedded Operating System

Trainer - Devendra Dhande

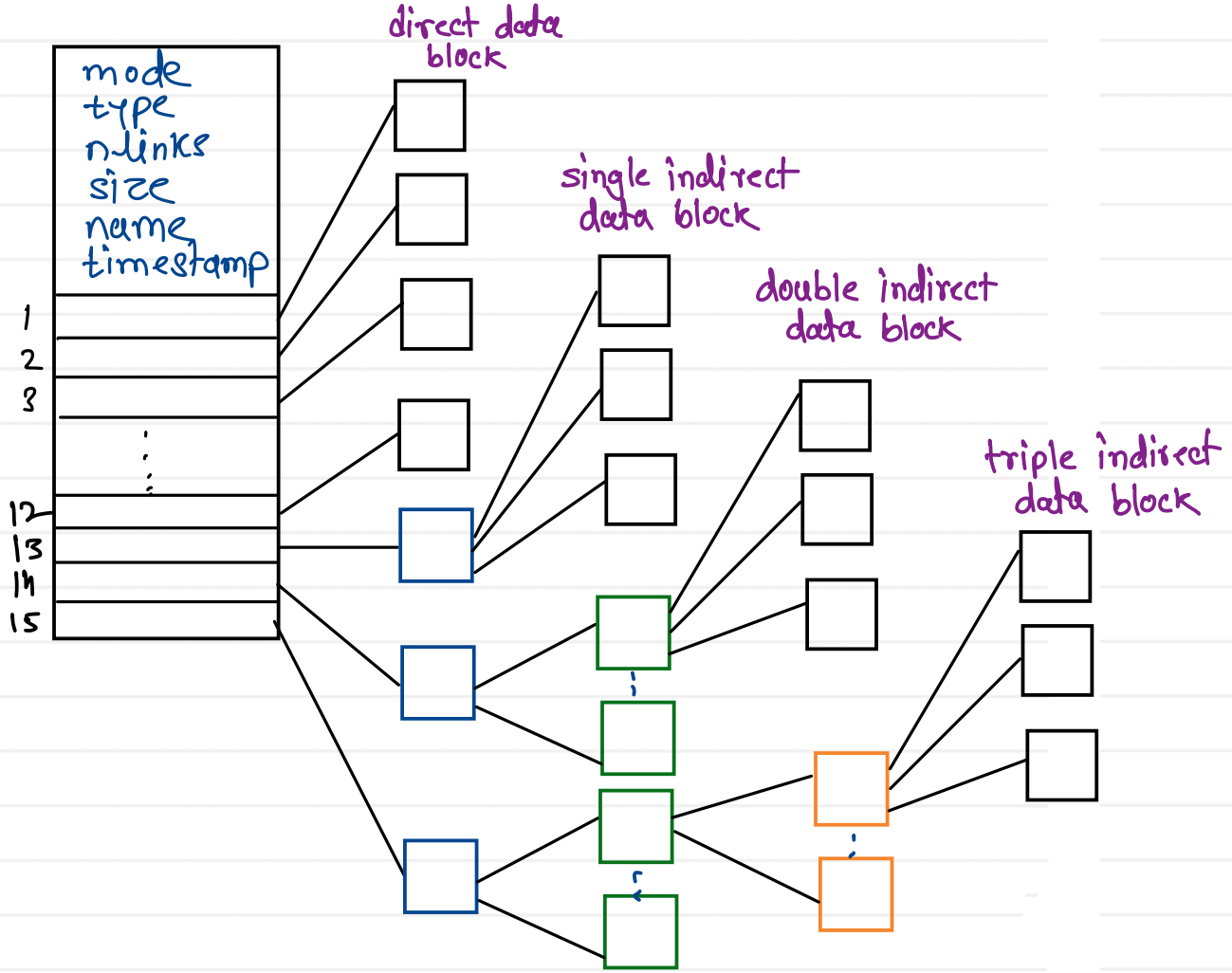
Email – devendra.dhande@sunbeaminfo.com

UFS (UNIX File System)

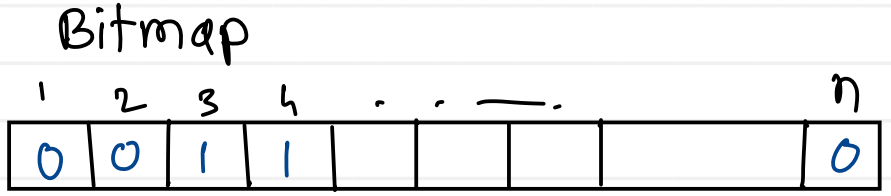




Ext file system

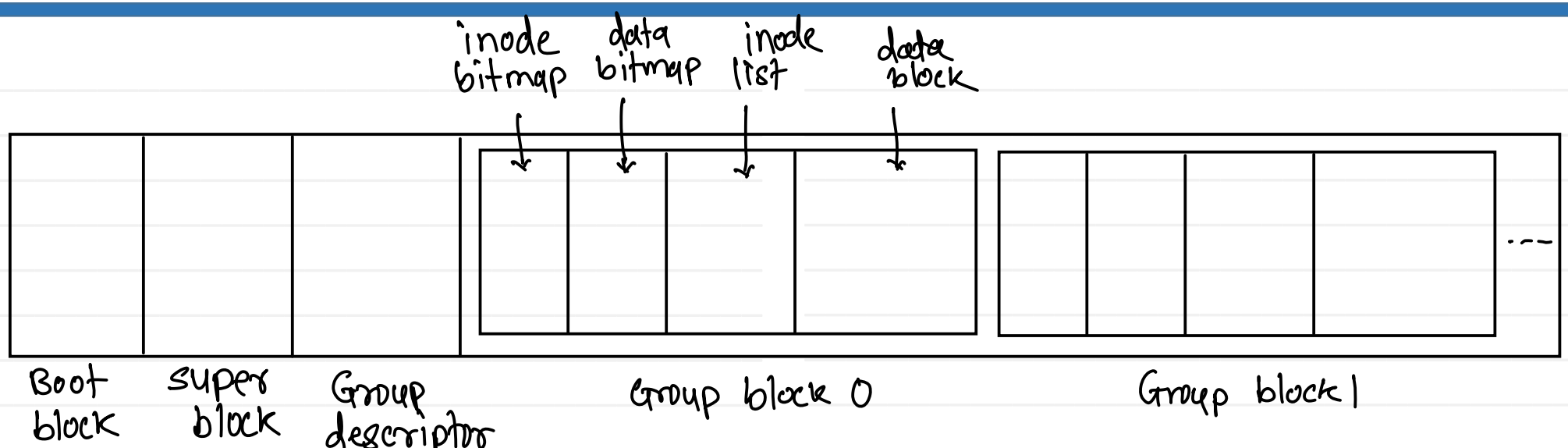


Indexed allocation



Free space management

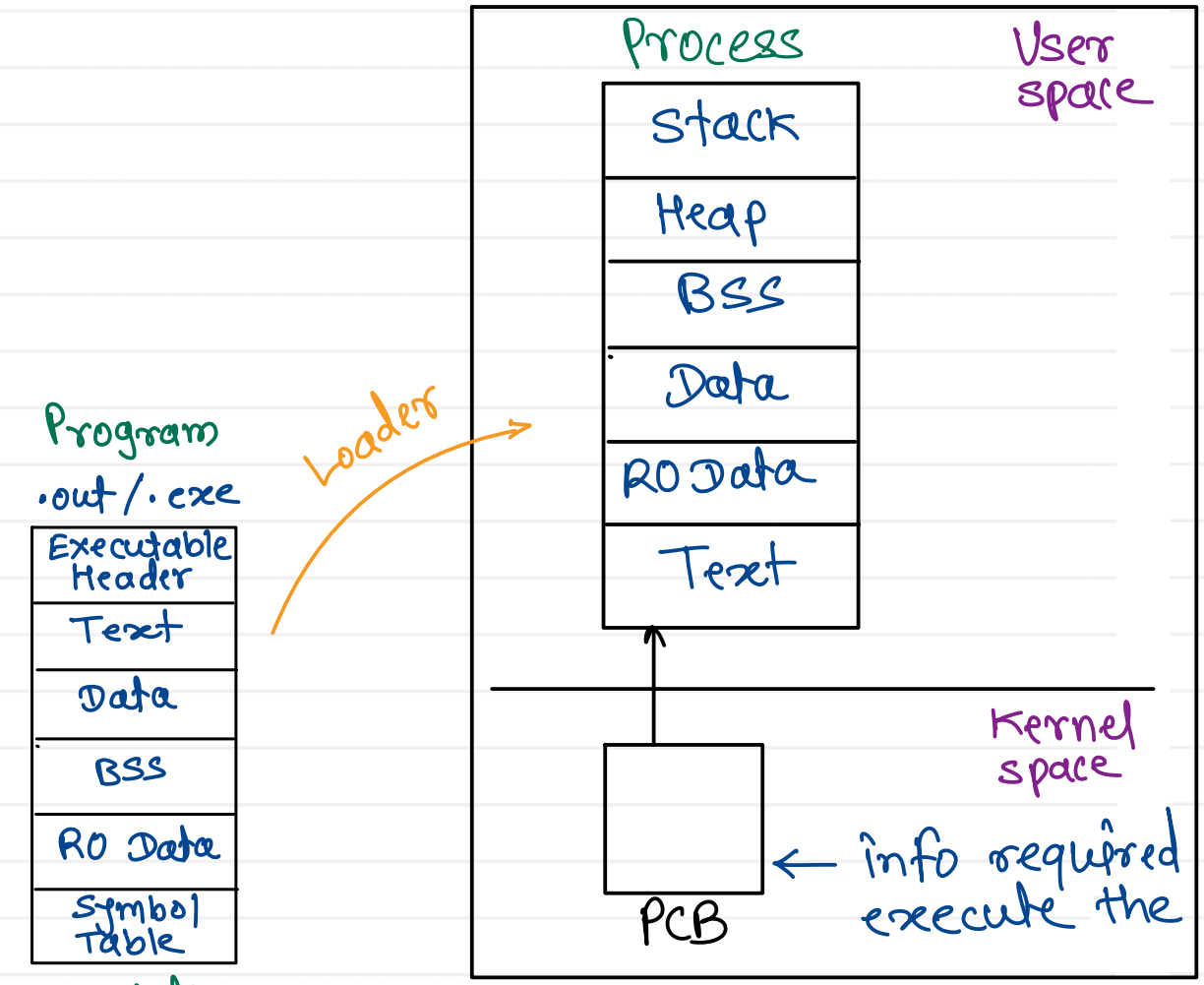
Ext filesystem



info about group blocks

file = dentry + inode + data block

Process



PCB - Process Control Block
↳ Process Descriptor
↓
struct task_struct
<sched.h>

UNIX - uarea

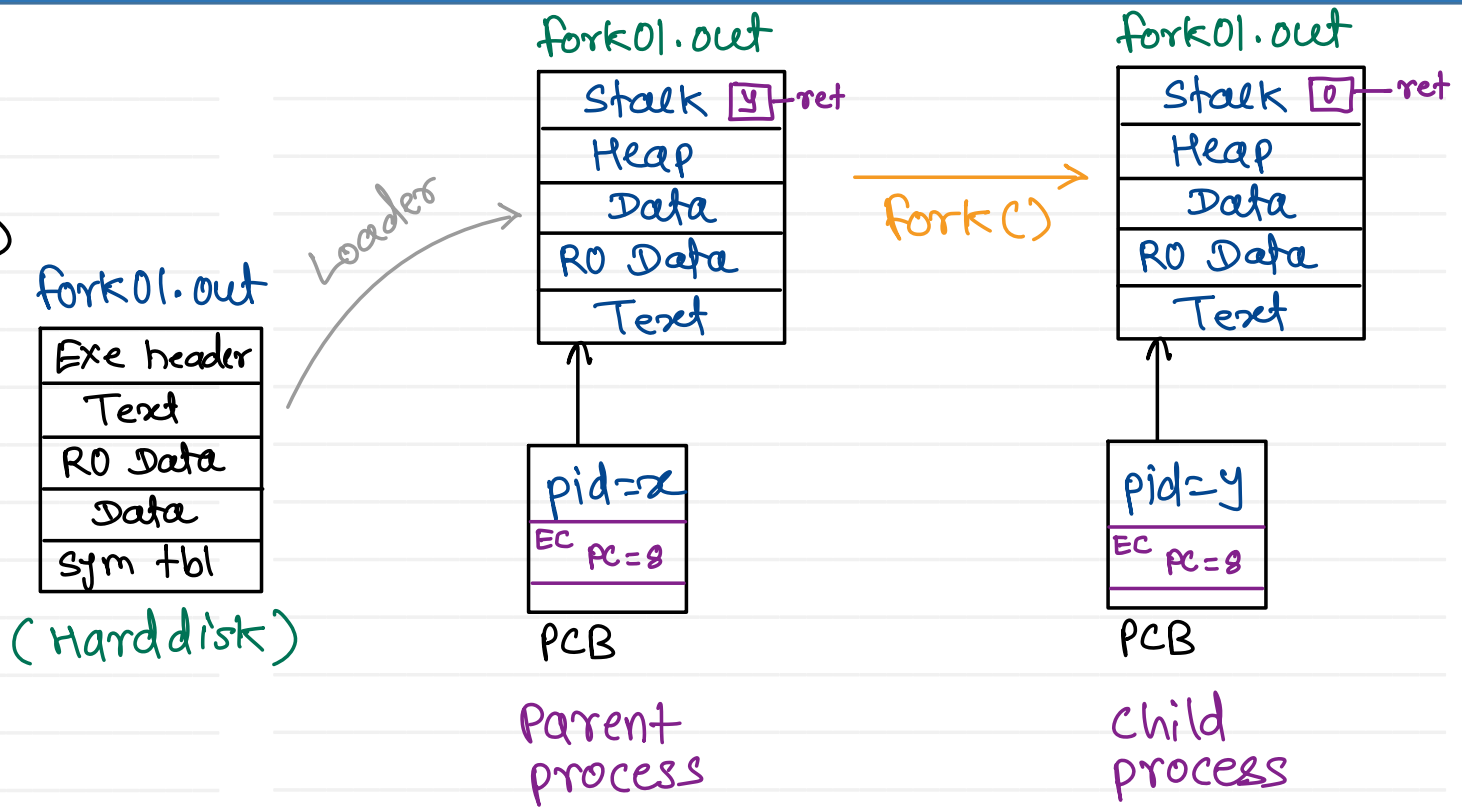
PCB :
1. pid, ppid
2. exit status
3. sched info (pri, state, algo .....)
4. mem info (base & limit, page/segment table)
5. file info (opened file, pwd .....)
6. IPC info (signal, buffers.. ——)
7. execution context
8. kernel stack

Process creation

1. Windows : create_process()
2. UNIX : fork()
3. BSD UNIX : fork(), vfork()
4. Linux : fork(), vfork(), clone()

```
#include<stdio.h>
#include<unistd.h>

int main(void) {
    printf("program started");
    fork();
    printf("program finished");
    return 0;
}
```



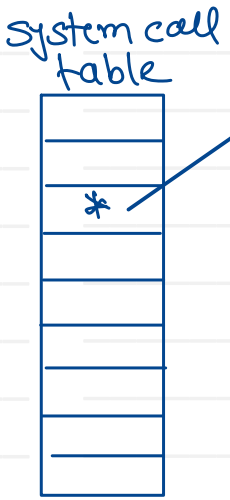
- child process is created by duplicating parent process.
- calling process = parent process, new process = child process
- both processes has seperate memory spaces.
- both processes are scheduled seperately.

fork() internals

```
int main(void) {
    int ret = fork();
    printf("ret = %d", ret);
    return 0;
}
```

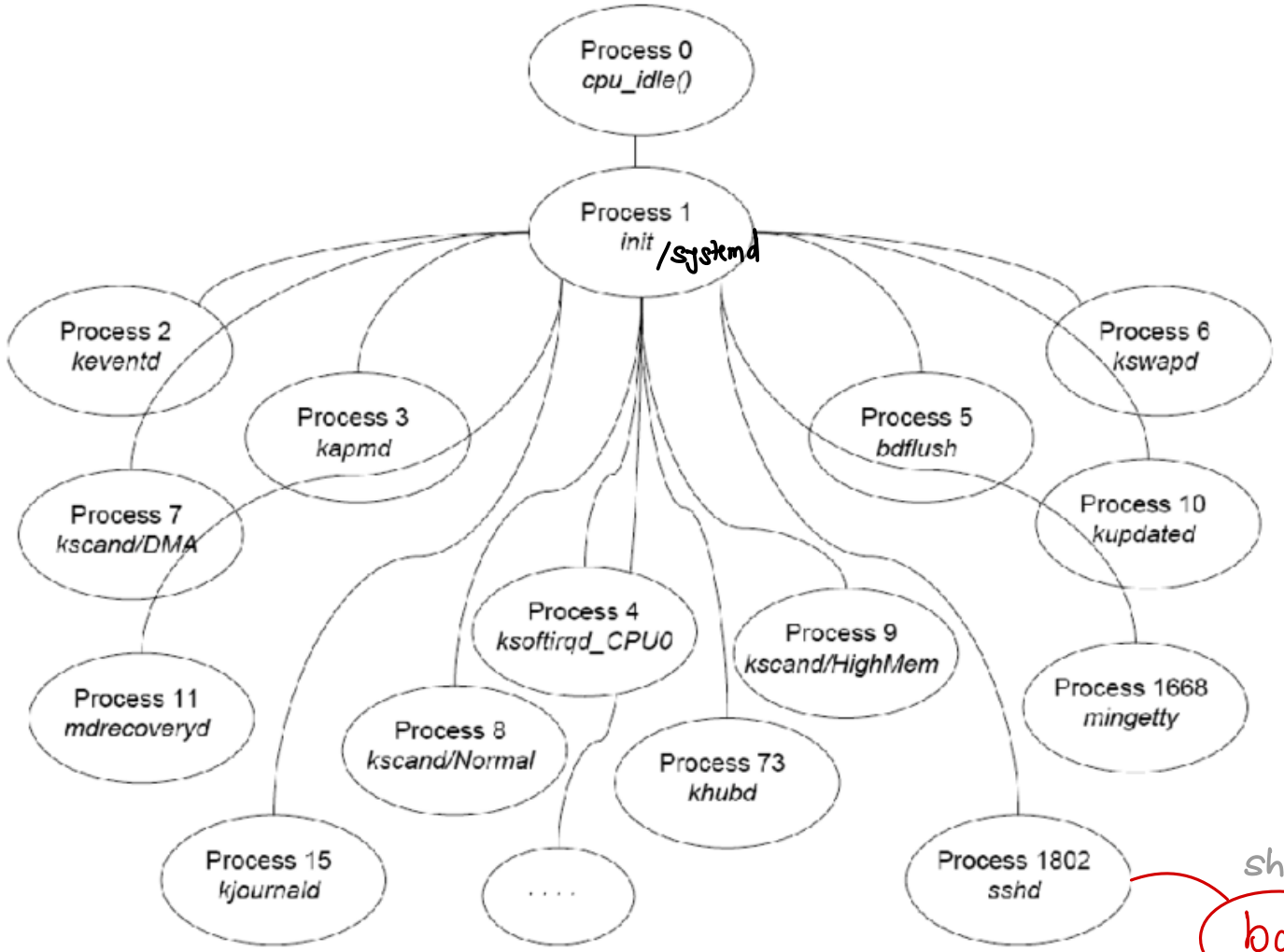
Return value of fork :
- On success :
  - PID of child is returned into parent
  - 0 is returned into child
- On failure :
  - -1 is returned into parent & child process is not created.

```
swi_handler( ) {
    1. save execution context of current process into kernel stack of process.
    2. find address of actual system call implementation from system call table
    3. call system call implementation
    4. pid = CPU_scheduler( );
    5. restore the execution of selected process from kernel stack on CPU.
}
```




```
sys_fork( ) {
    1. allocate space for PCB of new process
    2. copy PCB of parent into child's PCB
    3. change few values from child's PCB (pid..)
    4. allocate memory in user space
    5. copy parent's data into child's space
    6. Update return value in both
       i. parent process - execution context
          r0 = pid of child
       ii. child process - execution context
          r0 = 0
}
```

Process hierarchy

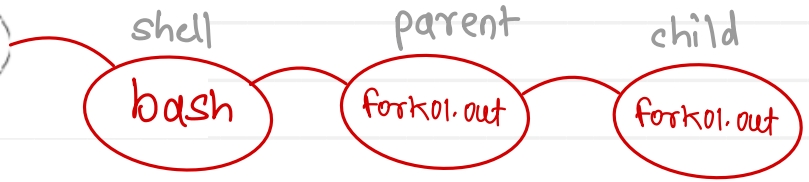


"init is first process created using fork(). It's pid is always 1.

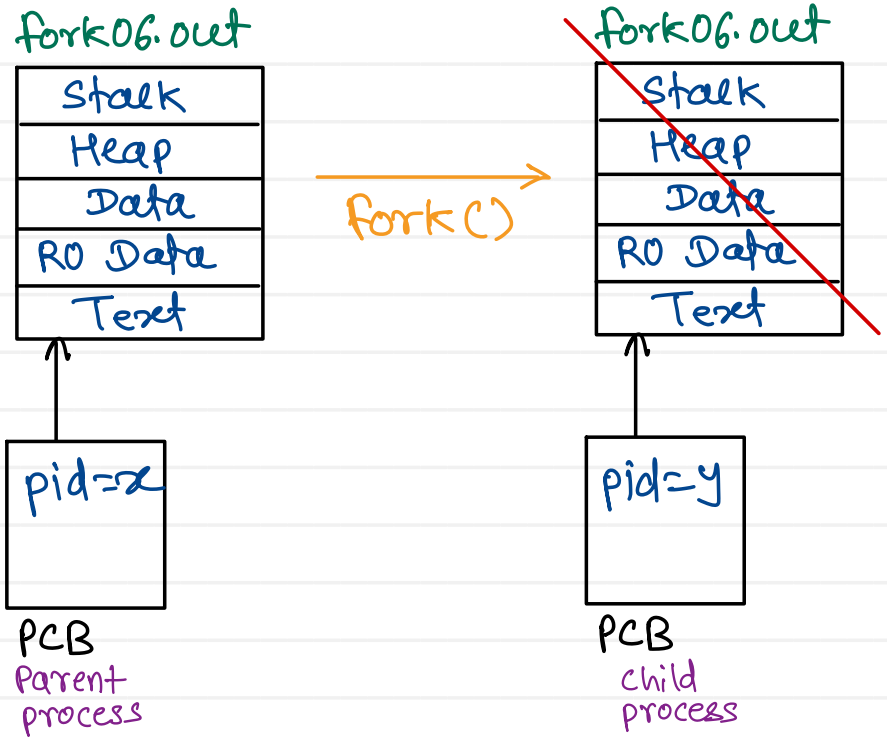
In modern linux kernels(3.0+), init is renamed to systemd and kernel initialization/startup is improved.

init was designed for uniprocessor systems, so services startup was sequential.

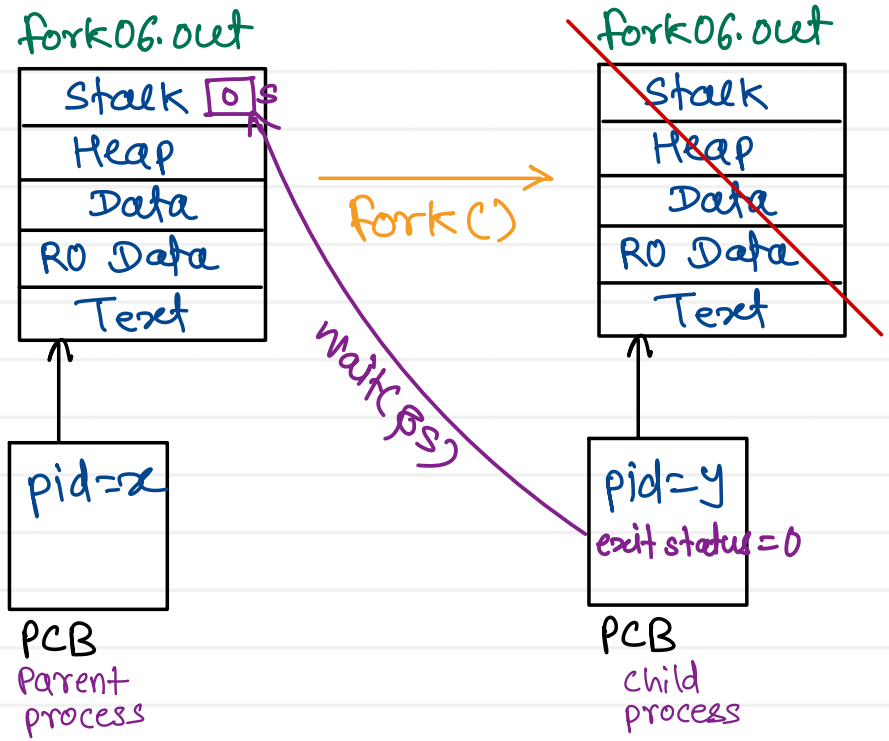
systemd designed for multiprocessor systems, so that multiple services can start simultaneously (can work on diff. CPUs). This speed up the system booting.



Zombie state



- when child is terminated before its parent & parent do not read exit status of child, child becomes zombie.
- PCB of zombie is kept into memory untill its parent read its exit status



- wait will block the execution of parent, will read exit status of child from its PCB & will store it into s variable. State of child will be changed to "DEAD".
- PCB of DEAD process is cleared immediately from RAM.

Thank you!!!

Devendra Dhande

devendra.dhande@sunbeaminfo.com