#6 LKF 2021

File operations

* File can be a directory, normal file, data stored in memory

/tmp → resides in RAM

* 

how it look in memory

* need a structure to organise data file system
  - Nfat
  - fat → ntfs
  - ext 2, 3, 4
  - xfs

metadata → using this locates data

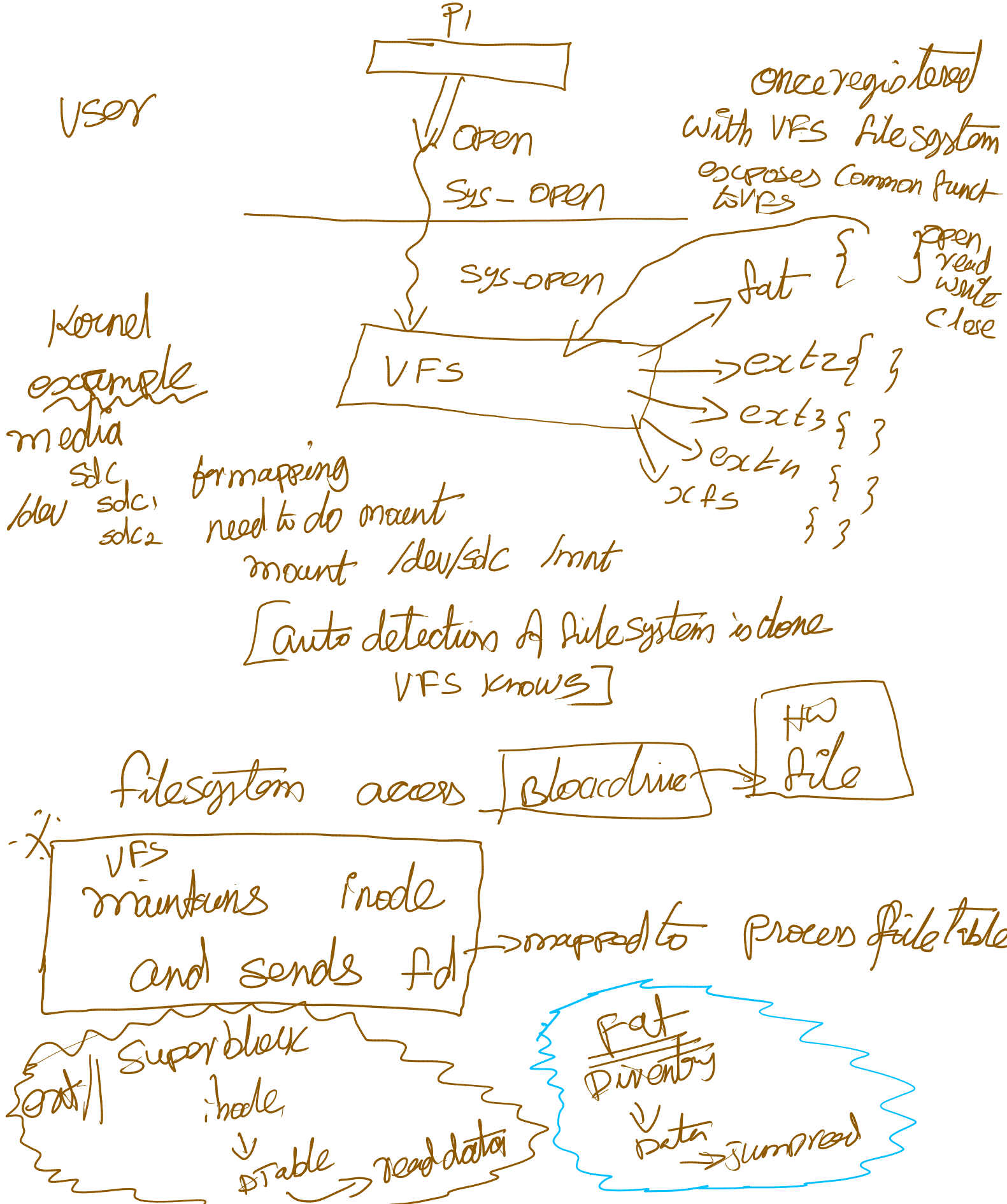
Data entry table then there will be link

1) Access of file go through in linux

P1

user

kernel


P1
User
open
Sys_open
Once registered with VFS filesystem exposes Common funct to VFS
sys_open
fat
ext2
ext3
ext4
xfs
open
read
write
close
VFS
Kernel
example
media
sdc
sdc1
sdc2
dev
for mapping
need to do mount
mount /dev/sdc /mnt
[auto detection of filesystem is done
VFS knows]
filesystem access
Blockdrive
HW
file
VFS
maintains inode
and sends fd
mapped to process file table
ext
superblock
inode
dtable
read data
fat
Directory
data
summary

Cat /proc/filesystems → VFS supported fs

1) first file operation is open

man 2 open

O_Create | O_Excl → to check file is present or not

Create function also returns fd

fd → has information about = { inode number, file operation structure, file pointer }

lseek → reposition the offset

seek_set, seek_cur, seek_end

Access permission → access(R, F);

R_OK, W_OK, X_OK

#7 LKF2021

*) dup and dup2 functions

*) Fcntl calls can be used for replicate dup dup2 redirection can be achieved.

*) MMAP → memory map

*) I/O Multiplexing

1) Select sys call → check for activity [if data available we can read no need to block]

2) Poll Sys Call

Normal file operations C library calls

FILE *
fopen
getc, putc
fgetsc, fputs

fscanf, fprintf
fread, fwrite
fclose, fflush, fseek, freopen

#8 LKF 2021

ftruncate (truncate syscall)

[Check for sufficient memory in file or disk]

examples: Code

a) fgets → read character till end of line
fread → read whole file depends on buff size.
no null termination.

fdopen ⇒ Converts fd to fptr
ftell ⇒ Current pointer
rewind → moves current pointer to zero

- fwrite
- fseek
- fprintf
- fscanf

hexdump -Cv file.txt

- Pread
- Pwrite

} Pass offset also as parameter
fptr is unaltered

Vector operation

- readv
- writev

scattered read or write

Makefile

Why makefile ?
dependencies of files in project need to be completed. [makefile has dependency graph].
aware of multithreading

make -f junkfile

make -C → Makefile is in different folder

```
test :
    echo "How are you doing"
    ls
    ps
```

```
all : run1 run2 run3
run1 :
    @ echo "run1"

run2 :
    @ echo "run2"

run3 : @ echo "run3"
```

#9 LKF 2021

makefile implicit variables

variables available in makefile

AR CPP

AS RM

CC

CXX

1) It has implicit compilation

2) make -d
   ↳ debug

Command line for makefile

make Hello=test

$@ → target

$< → prerequisits

$?

$^

$|

$*

3) makefile Variables

handles unknown behaviour

4) Wild Card in makefile → $(wildcard *.c)

SHELL → we can modify shell

Variables

=

:=

+=

?=

test = var1

test2 := var1

test3 ?= var1

test4 += var1

test:

echo $(test1)

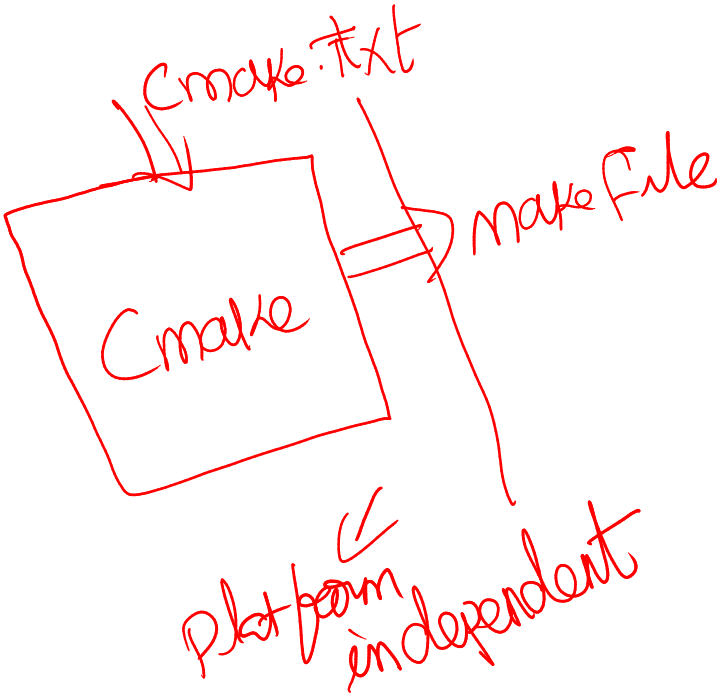
1) override command line argument

2) PHONY → .PHONY : clean

3) DELETE_ON_ERROR

4) foreach

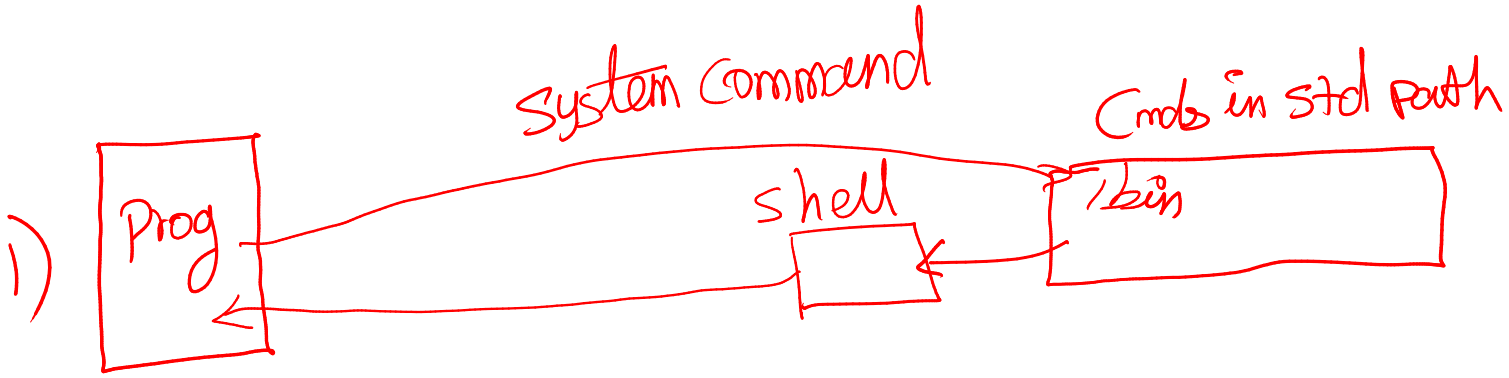
5) Strip

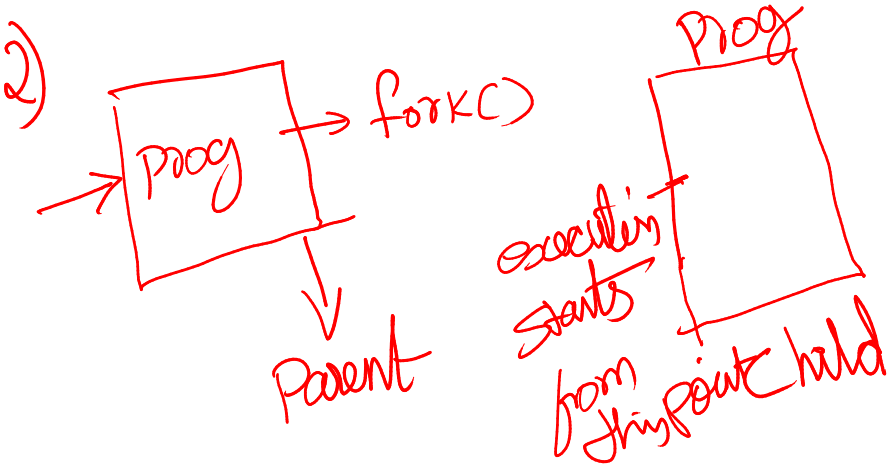


#10 LKF 2021

Process Management

1) 3 way of creating process
[System, fork, exec]

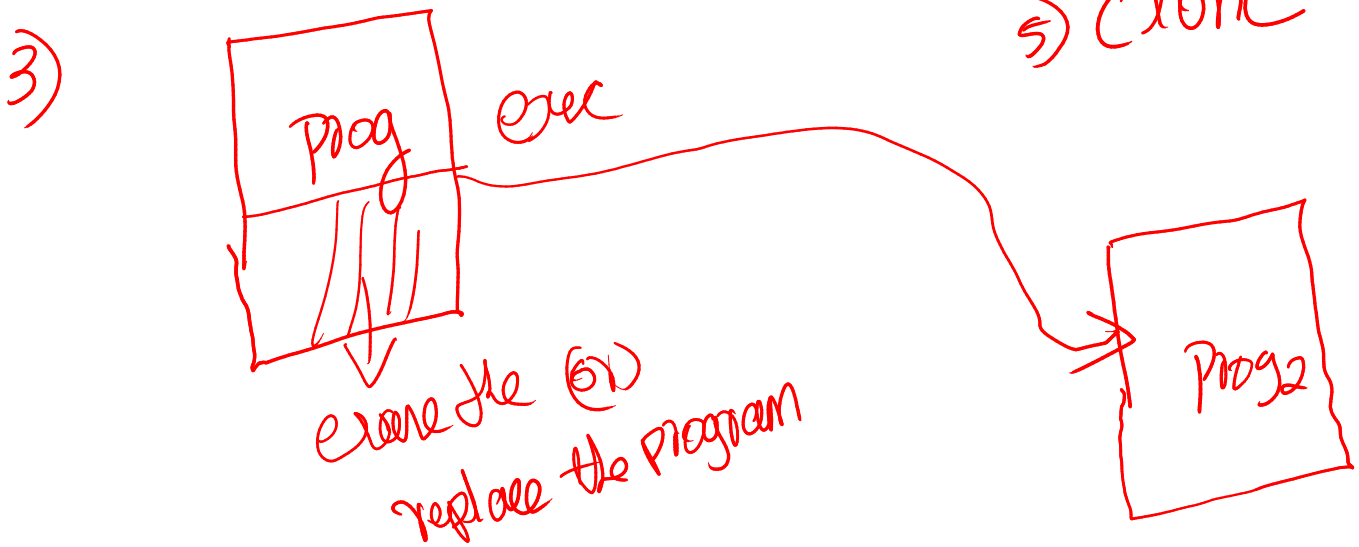




Exact copy made

4) Vfork

5) clone



System Command

int system (const char* Command)

→ this spawn shell execute the Command

block till Command executes

Examples:

system ("ls -l");

Check return value

echo $?

rax hold return value

To prove it spawn shell give blocking cmd to system

(cat, top, tail -f)

fork() Call

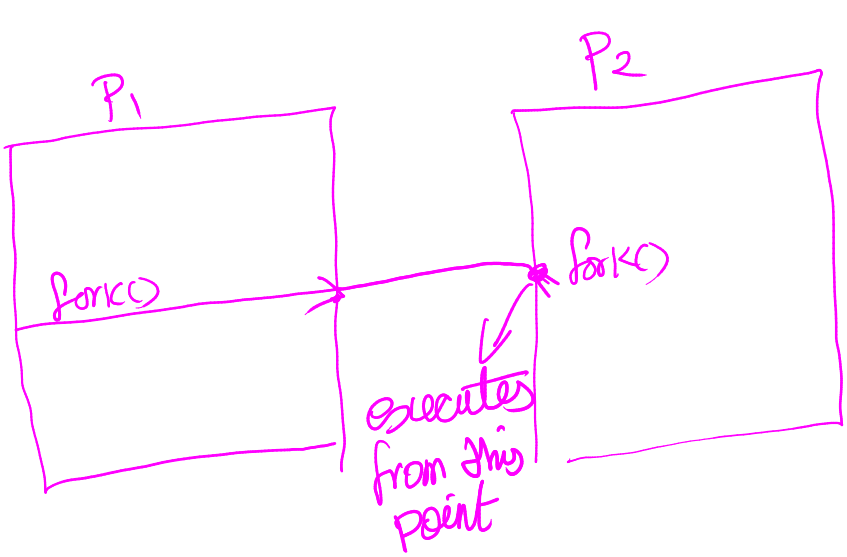
Pid_t fork (void)

0, PID of child, -1

By return value we can find Parent or child.

Program text
Stack
PCB
Data

→ all are copied to child process



COW
Copy on write

PCB ⇒ Process Control Block
(or)
TCB

Example:

Program:

1) Comment sleep()
put sleep() in Parent
} Create Zombie Process

Parent responsible to clean all resources

two ways

wait call

exit of Parent
(or)
kill Parent

2) Orphan Process

system deamon will become parent to that child

[systemd has capability to remove (or) clean the resources]

*) Change variable in both process and check

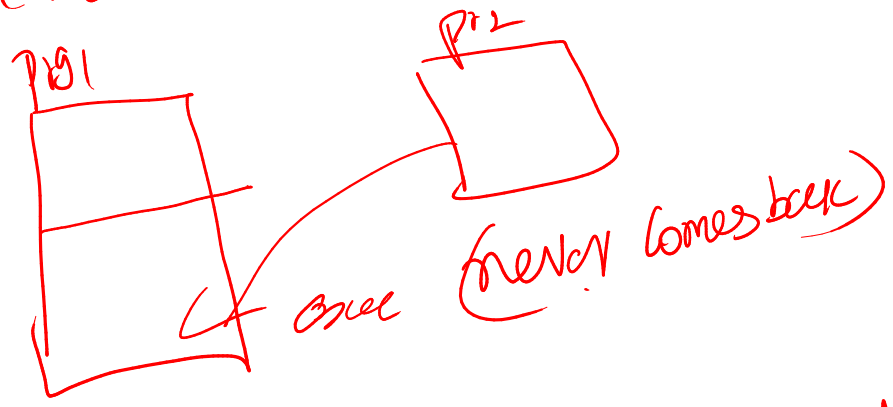
*) fork();
fork();
2^n times ie 2^2 = 4 process

rare case we will do this

*) file reading

debugger
set detach-on-fork off
show detach-on-fork

Exec call:
Where you want to change the complete behaviour of the program.



[Program will be replaced]

examples:
execl => Complete path required
execlp => no need complete path
execv => need path, argument in array
execvp => no need path, argument in array

Practical ex
1) # ls

↓ child process shell

2) client server example

Vfork() [careful in Vfork Programming]

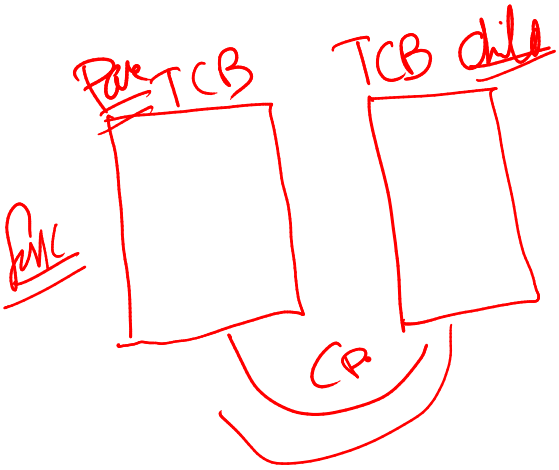
fork() → Time Consuming process [not parallel]

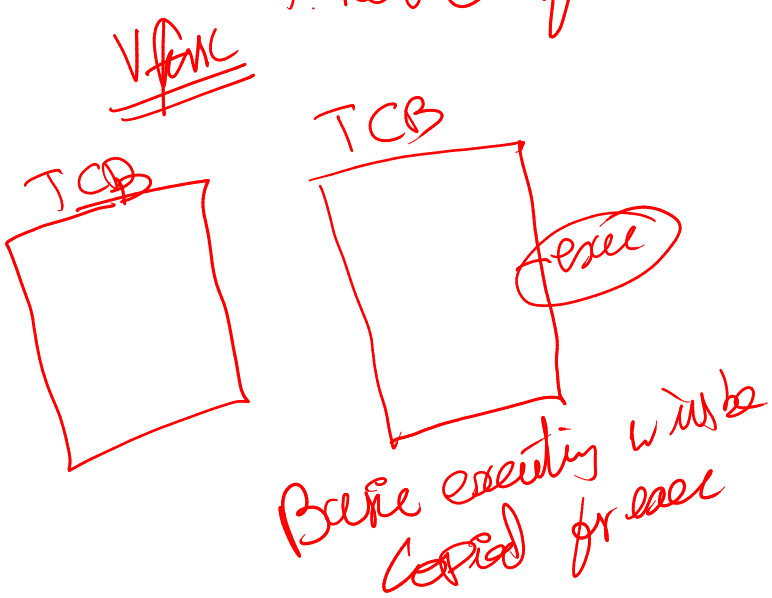
↓

1) Child & Parent share same address space

2) Child runs first then Parent runs.

*need clarification





Before executing will be copied for exec

Example Programs:

1) segmentation fault

[Child process should exit with exit 0]

will give segmentation fault

* Parent will wait till child exits ie blocks Parent

Variable printing test example:

Variable modified in child reflect in parent

Clone:

Superset of all

fork
vfork
thread
} everything calls clone.

man clone

example clone:

- ./clone
- ./clone Vm

PID

getPid()
getPPid()

exit call

voluntary removal resources
i.e detach the resources from process.

#11 LKF 2021

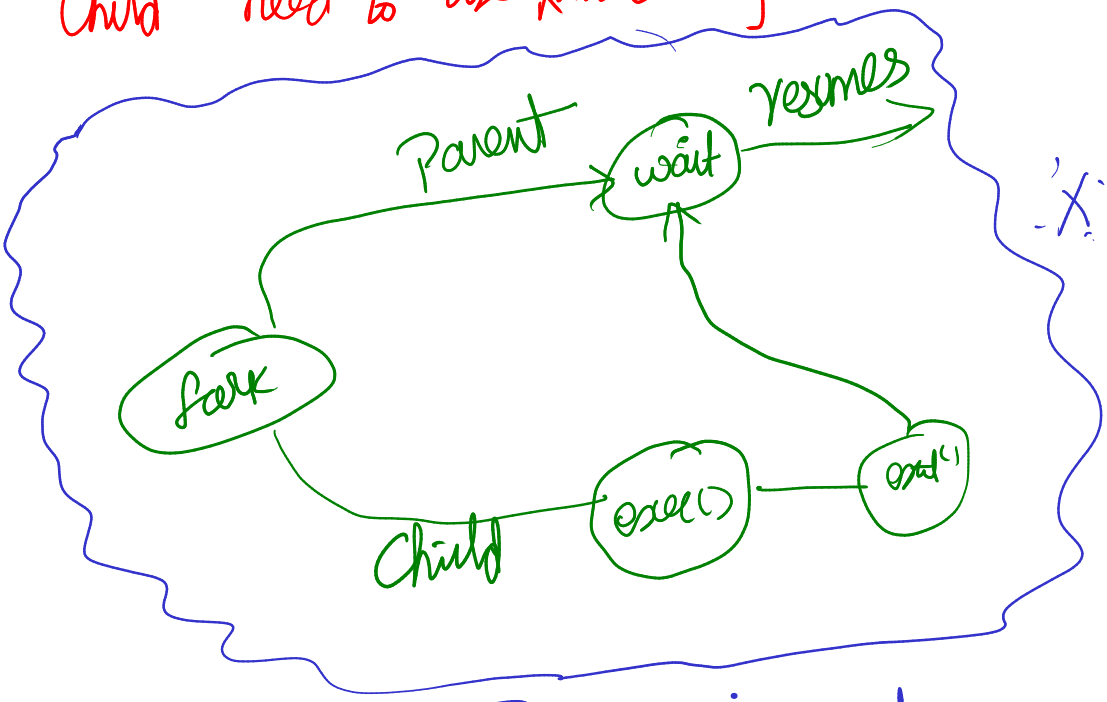
Wait calls:

wait() waitpid()

Wait for
to childs

wait()

{to get info of exit status of child need to use defined macro}



examples:

- child never exist parent also never exist

(((int)0) = 0

Killed by signal 11

assigning value 0 to address 0

Threads:

*) Process creation is costly
*) need simple mechanism that they created thread
↓
light weight process

lot of critical section & race condition need to take care.

:(main program should wait for thread

Pthread_join
Pthread_exit } use this

Examples:

thread → stacks can be accessed by other thread using &.

TLS
Thread Local Storage

*) Pthread_join → call in main process and we can call in thread also.
*) Pthread_self → to get TID of thread.

IPC

1) Shared memory
2) mapped memory
3) Pipe (Named & unnamed)
4) Socket [RPC]

1) Pipe
access is only in parent & child process only

#12 LKF 2021

IPC

2) Pipes in deep through example

popen → execute a cmd, data will come to file pointer

[not a blocking call]

PClose

1) system ("ls > /tmp/lsoutput.txt")

   openfile
   read

2) ope file "out.txt"

   dup(xx, stdout);
   system command

Named Pipes:

mkfifo:

example: