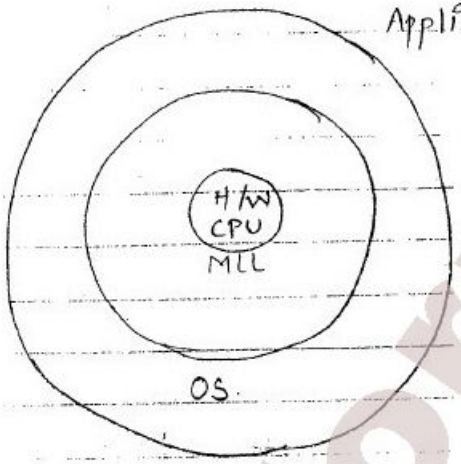
OPERATING SYSTEM SECURITY

→ INTRODUCTION
→ Mem/ address space protection
→ Protecting general object
→ Protecting file
→ User Authentications

INTRODUCTION :-

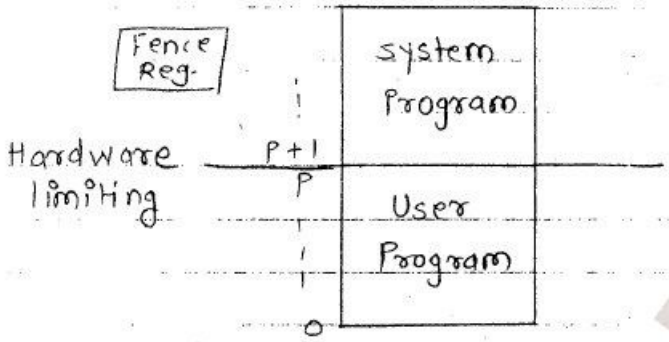


1] Executives :- small / simple function. to conver HLL to MLL

2] Monitors :- some more complicated funct?

MEMORY / ADDRESS SPACE PROTECTION :-

1] FENCE :- (CONFINMENT) :-



(SINGLE USER)

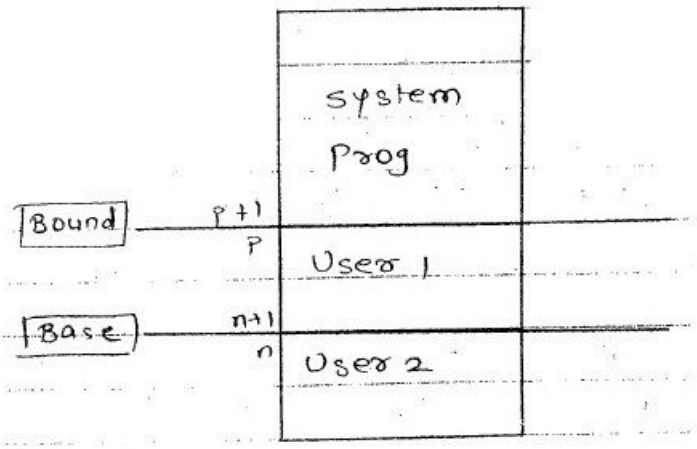
(O.S size should be fixed)

Limitation :-

If system prog or user prog some more space the whole hardware should be changed therefore s/w came into a picture which is called as Fence Register.

Limitation of a Fence Register :-

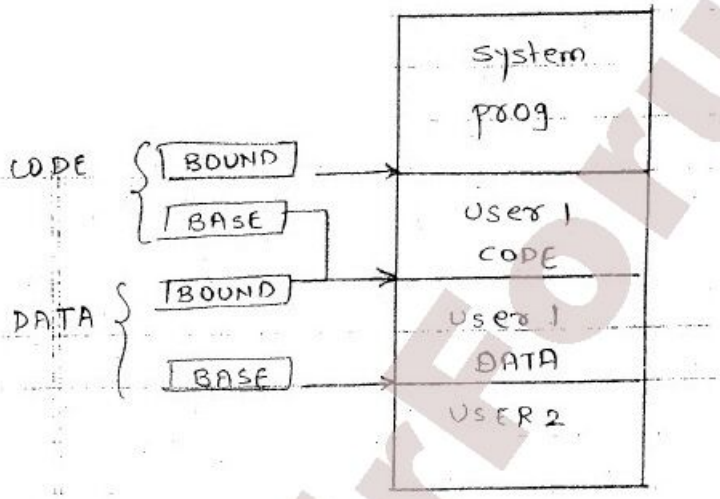
It can be used only in case of a single user.



II] BASE - BOUND REG :-

It is used in case if user is more than 1

III] PAIR OF BASE - BOUND REG :-



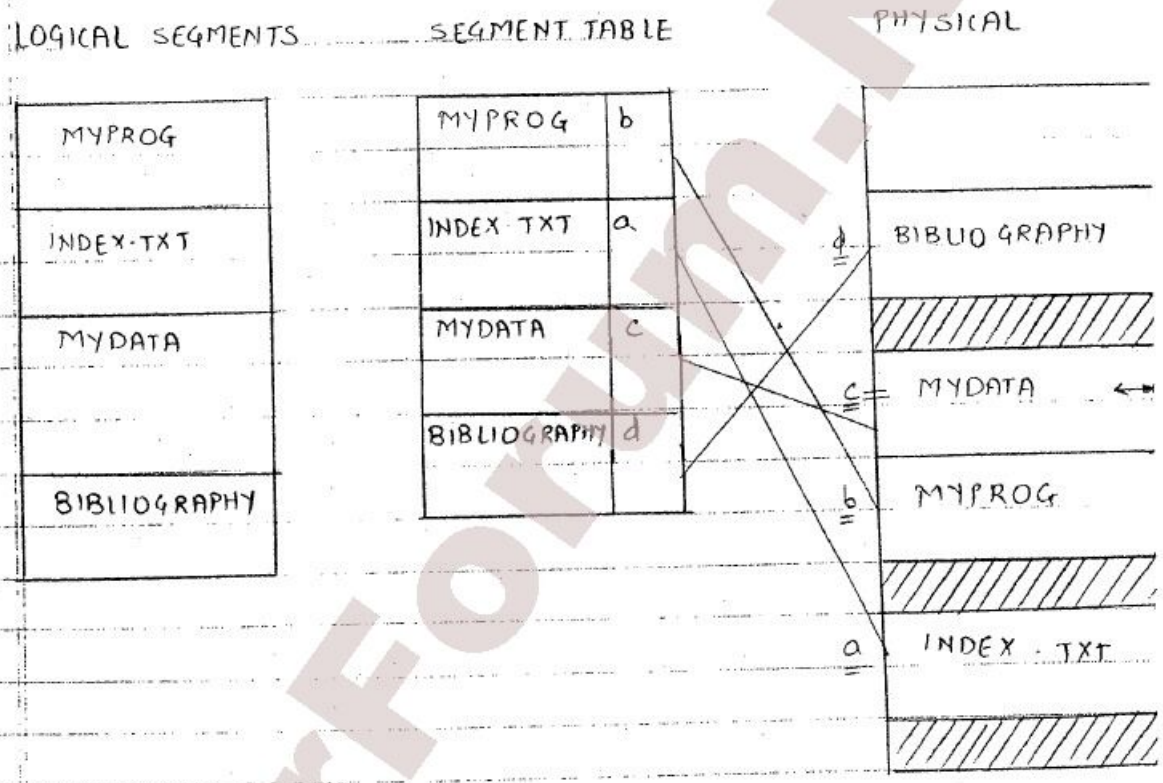
Context switch came into picture for reducing the no of register.

→ values of Base Bound reg changes according to the next user.

IV] TAGGED ARCHITECTURE :-

TAG	ADDRESS	CONTENTS
R	3000	3E
R, M	3001	49
X	3002	5C
R	3003	4C

SEGMENTATION :-



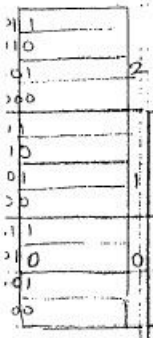
→ segmentation is variable size
→ logical entity
→ Fragmentation outside

User Access

< seg. Name, offset >
Mydata, 35

→ Vulnerable to overflow
→ Assigning attributes or tags is easy.
→ Proper Authentication since every access through C

PAGING :-



Page 1
Page 2
Page 3
Page 4
Page 5
Page 6
Page 7

Page	
1	h
2	e
3	J
4	d
5	g
6	a
7	b

a	6
b	7
c	
d	4
e	2
f	
g	5
h	1
i	
j	3
k	

user access
< page no, offset >
2 , 7

Base 1000
offset + 0111
1111
↳ overflow (Page 3)

my assume every page is of 4 byte.

→ Paging is of fixed size
→ Physical entity
→ Fragmentation is inside
→ overflow is detected
→ Dynamic allocation of memory is not possible in paging since the page is of fixed size.
→ Assigning attributes to the page is not easy
→ Proper Authentication since every access through o.s

CONCLUSION :-

From security point of view paged segmentation gives maximum protection to the address space

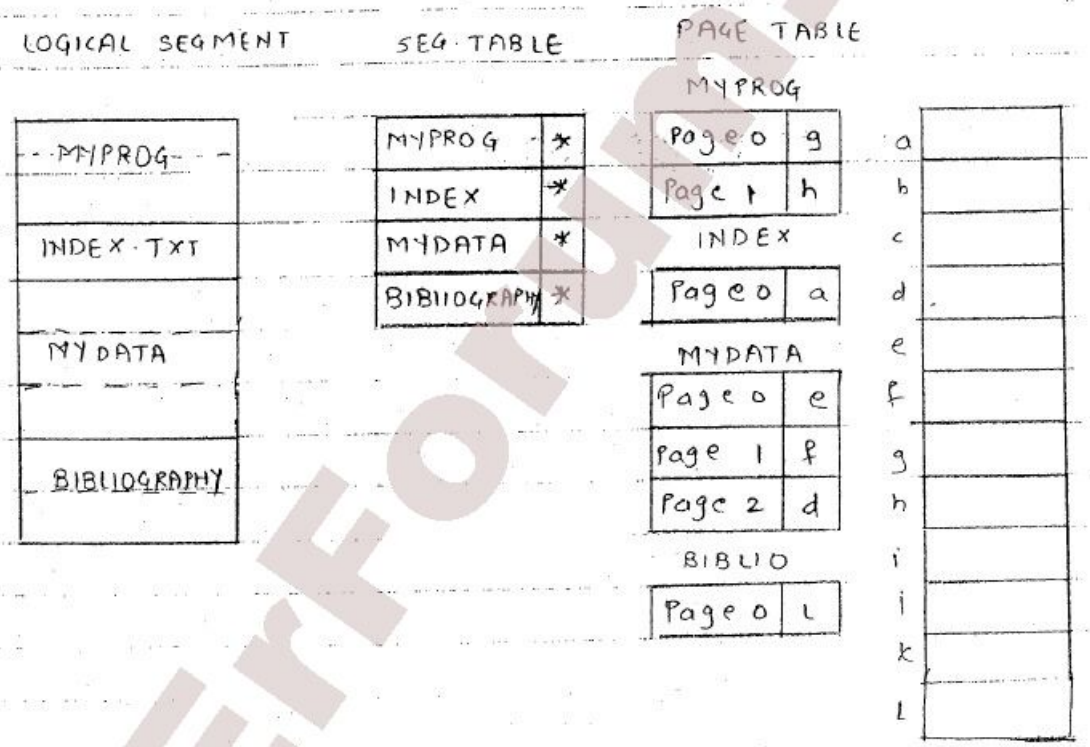


FIG :- PAGED SEGMENTATION :-

PROTECTING GENERAL OBJECT :-

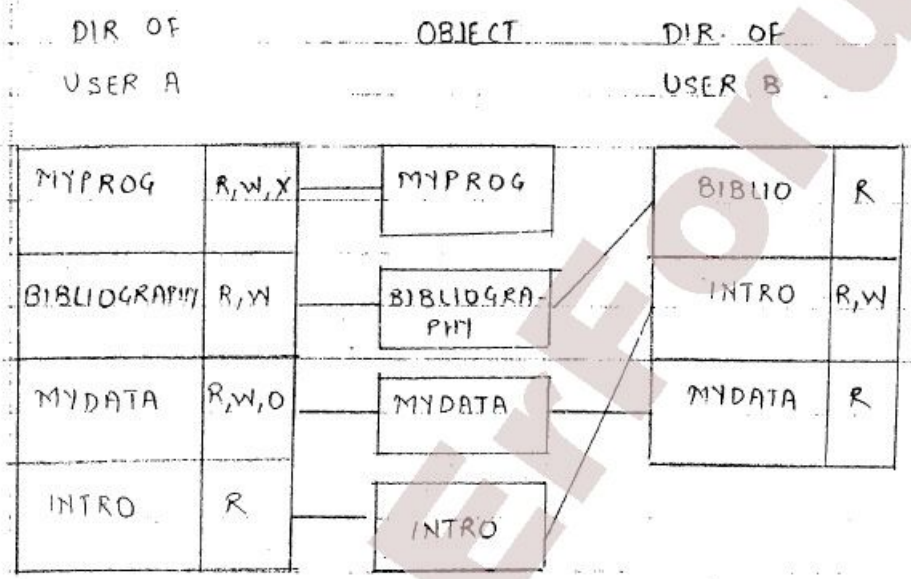
OBJECT :-

Database
Program
Device Driver
file
|

SHARABLE OBJECT :-

Library routines.

1] DIRECTORY METHOD :-



DIFFICULTIES :-

1] DIRectory have large no of entries in case of sharable objects.

2] REVOKING THE ACCESS :-

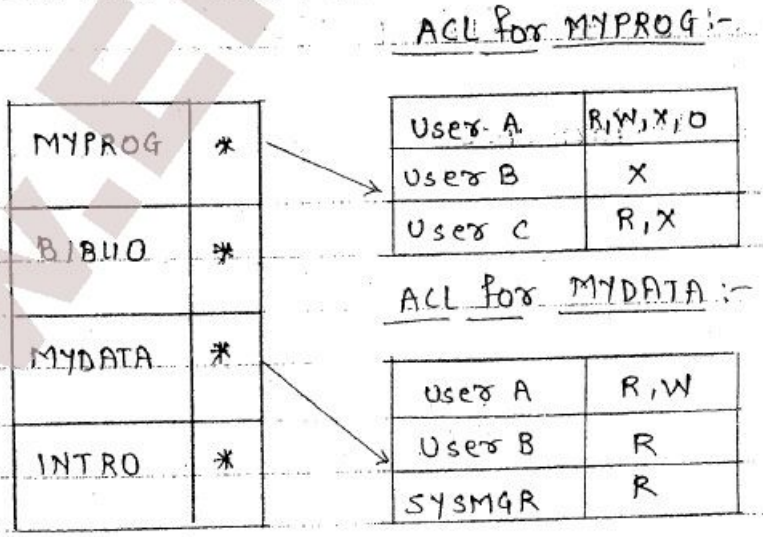
If process A & B is having trusting relationship, the objects of A & B are sharable. But after sometime A & B relationship is over A has to revoke the object from B.

3] PSEUDONYMS :-

Every process should have objects with unique name. for eg:- If A is having MYPROG object as another process C is having & if now B wants the access of MYPROG object then there will be a problem of saving this object into process B. so that B will ~~not~~ not know from which process the object comes.

2] ACCESS CONTROL LIST (ACL) :-

In Directory Method O.S was having the process with object names of which authority the process has. In Access control list O.S. will create a list of objects that which process are using those objects.



< user, group, Compartment >

WILD CARD ACCESS :-

In wild card entry all those objects which ar

being accessed by any process are saved.
It is also called as star access.

3] ACCESS CONTROL MATRIX :-

	User A	User B	User C
BIBLIO	R	R, W	—
INTRO	—	R	—
MYDATA	R	—	—
MYPROG	—	—	R

< subject , object , Access Right >

It is called as SPARSE Matrix.

- Poor Performance

4) Capability : Up till now only OS Keeps track of all protection objects & rights. here we put some Burden on user. User may require to have ticket or pass that enables access, which cannot be duplicated User can create new object. & also completely new data object
A capability is a ticket giving permission to a subject to have a certain type of access to an object. to offer solid protection the ticket must be unforgeable. The OS holds all ticket on behalf of users. One possible access right to an object is transfer or propagate

Basic forms of Protection :- → All-None Protection fails → Lacks flexibility
→ allow nothing
→ Rise of time sharing complexity
→ File listings.

→ Group protection :- → User, group, world } 3 classes.
group affiliation ↔
Multiple personality ↔
→ Single permission → Password / other token.

PROTECTING FILE

→ Loss
→ use
→ disclosure
→ accessed by each user.

USER AUTHENTICATION :-

[PASSWORD
BIOMETRICS]

User can be authenticated to the files by 3 different ways :-

WHAT USER KNOWS :-

It is used as password. (Names / places)

WHAT USER HAS :-

It is used as password (Licence / PAN / ID)

WHAT USER IS :-

It is used as Biometrics. (Finger Prints / Retina / Voice sample

ATTACKS ON PASSWORD :-

1] Try all combinations 2) Try many probable password 3) Try password likely for the user 4) Search for system list of passwords. 5) ask user.

① — Brute force attack
— Exaustive attack

→ A-Z all 26 combinations ie :- $26 + 26^2 + 26^3 + \dots 26^8$ password of 8 characters
→ 5×10^{12} or 5 million will take 150 years if one password / millisecond.

② Probable password : $26^1 + 26^2 + 26^3 = 18278$ of length 3 or less.
③ Password likely for user.
④ Plain text system Password list : Target system Password file
⑤ Encrypt password.
⑥ Indiscrete user (Social Engineering)

Guidlines for selecting a password:-

select large size password
changing password regularly
Alphanumerical password.
Never reveal the password to anyone.
One time password.

2] Try all probable combinations.
Dictionary based password

3] Try all possible combinations for user.

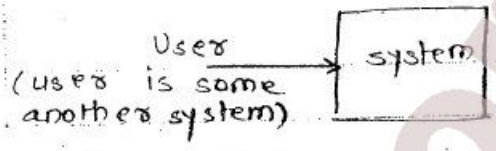
4] social Engineering

USER AUTHENTICATION:-

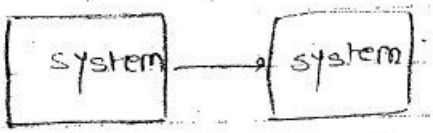
password Attacks
Guidlines for password selection
one time password.

ONE-TIME PASSWORD:-

system Authentication



user of system is agreed on some function
say $f(x) = x^2 + 2$
user gives some challenge to system
everytime password is changing in response to challenge.
$f(x)$ may be complex ∴ Every time to calculate $f(x)$ for given i/p is very complex
∴ there will be another system instead of user.



LOOSE LIPPED SYSTEM :-

Example :- 1

In case of login : welcome to xyz company
username : ADAM
If username is wrong then system respond here only than invalid user
system again ask for username Hll correct name is entered & then ask for password.

Example :- 2

welcome to XYZ
USERNAM :- ADAM : whether wrong or right it asks for password.
password :-
If wrong one of both give message invalid Access

Incase 2 :- User don't know whether user name or password is wrong where as in case 1 user can know the user name is invalid.

case 3 :- username :
password :

If authorized user the message "welcome xyz company"

User can't even know name of company unless he is valid user.

case 1 - case 3

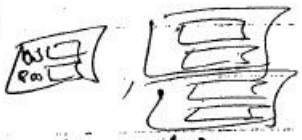
ENCRYPTED SYSTEM PASSWORD FILE :-

user tries to login to system & it checks for password for checking password system maintains

---

★ USER AUTHENTICATION

→ Something User knows → User has → User is



USE OF PASSWORD → Loose-Lipped-System
→ Additional authentication information

★ Authentication process :
Four flaws in authentication process

(i) Challenge Response Systems
(ii) Impersonation of login (Trojan horse)

Authentication other than Password
→ Sophisticated authentication device.
eg: Fingerprint reader, voice recognizer, retina pattern recognizer