

Session 1, Adv

* الراجل كتب الكتاب عن طريق أنه يقرأ الـ doc لوجودة، ويحمل برنامج صيغة test، بعد ذلك ما تفهم الـ Concept كوس بعد كده كيش واكتب Code وريلف طريقة لفهم الـ Code Concepts *

* Linux بناء على history، وكانت AT&T كانت تختبر زمان وطالع المحكمة الأمريكية tele-communication، قرار فحقو اشتغل في court في الـ SW لانها محتكرة، فقالوا لها مستخلص في الـ SW

* جامعت berkeley زوروا فيها Features من UNIX Versions، متى طلعوا من الـ

* AT&T الحكومية كانت وعندم شركات berkeley ورفضت، وكانت قليل عيدها 18 files، وكان في 3 files كانت berkeley ادخلت UNIX في المكانت، وكانتBSD

Look @ email at page 51

Standardization (page 54) :-

الناس خلقت الـ C زورت كلها كما أنها فتحت
برهان مثال على programs في اجهزة
واجهزة لا

الناس يأخذون مع Compute standards ويعملون
في الـ autoCAD implementation

C standardization :-

C99 C89 كـ ANSI C زوال

POSIX standardizations :-

Portable operating اختصار POSIX الـ
system interface unix

interface، الاختصار POSIX الـ standard هو POSIX
مع الـ operating systems، يساعد على
الـ interfacing مع الـ OS لأنه يكون
بشكل واحد لكن implementations لا خاص بكل
Vendor

ـ IEEE، ولو انت حاصل تزور حاجة
ـ Free Cuis برامج و هو POSIX في Cuis

FIPS :-

*الحكومة الأمريكية كمان وقالت
ان اي حاسوب يكون Computer لازم يكون
مكالم مع الـ FIPS و مات مكالم

Federal Information Processing Standards

SVS (Page 57) :-

* في حين يدعى بـ بروكات و الـ open group .
نادي كالـ Unix like OS Standard calls 4 Parts of os Standard الـ صفحة 3700

Base definitions
پیش تعیینات

System Interfaces:

Library Func. JI و Sys calls بيكفال

Shell & utilites:-

utilites وحة ازاي شل Shell

Look @ Page 61

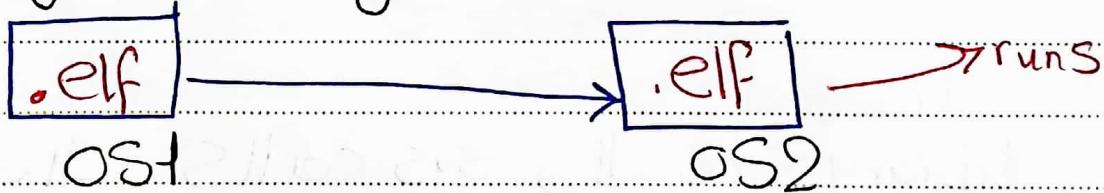
Linux Standard Base (LSB) «Page 63»

* انت كنيلك الراجل اللي اسماه Linus Torvalds maintain Kernel حالياً

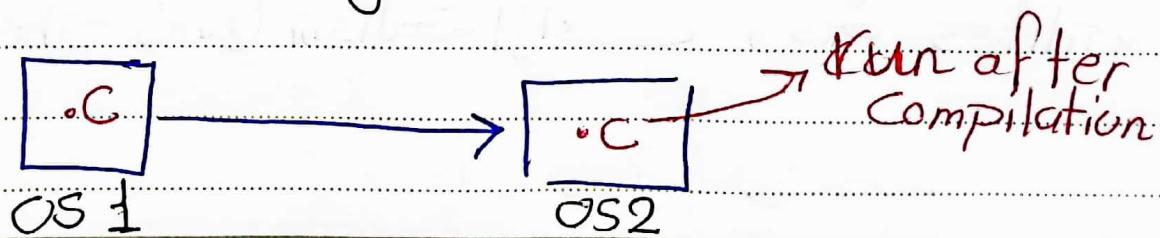
* اطربتني ايه اللي يفهمنيلك انك لو اخترت run on Ubuntu على سخال binary على LSB أو Kali على

* ما لا binary Portability على LSB بتطلب interface POSIX وال SUS ال اقدر اخذ ال Code و احمل binary على Computer على OS او OS تاني ويطلع له يشتعل تاني

binary portability :-



interface Portability :-



* Chapter „2“ *

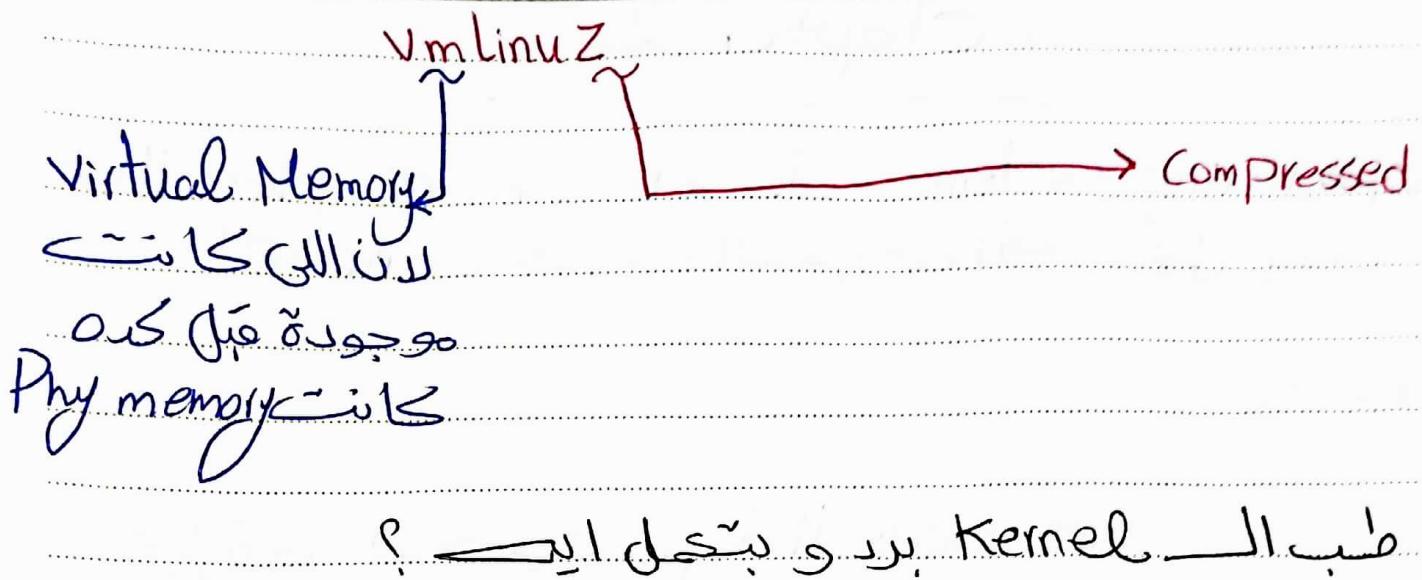
Kernel

Note:

Kernel Computer run in Embedded Computer
Manage bare metal System bare metal App

* طب ال Kernel يتحمل ايه ؟ يتسلل كتابات البرامج
→ Power Application جلسات بحسب متطلبات ال Application
→ Program ال Flexability اى

/boot/vmlinuz (指向 kernel executable) *



A-Process Scheduling:

في الـ switch بين المـ tasks وواد كانوا Linux non-preemptive او preemptive حال وخطالك ميفحدين الفرق ينفهم في اخر الـ PDF بتتابع السيشن بعـ

B - Memory Management -

ال Kernel RAM في المسؤولية عن التحكم في الـ limited resource التي هي بالمنا في طرق اسفلها يتضمن كل process كـ virtual Memory في process خاص بها و كل ا- يتعرض الى processes عن بعدهن كـ ثان معين process تكتب في الـ hardware يتبع التالية

ـ يتخلى الـ process مسؤولية عن حجم الـ RAM لـ cache حسنا وانا خاطر عندي انه ملكيتش رحوة انا في بعثت hard disk N pages واريكي الحسنا

C- Provision of File Systems

الـ hard disk هو عبارة عن حبة حديقة، فهو الذي يحتللى قارئ افتح واعدل واعمل وامسح **System File**

D- Creation & termination of processes

انت في الـ Linux متقدرش (process) create terminate
process create دخل او تدخل terminate
جديه، انت بترفع تدروه fork() sysCall او exit() و Kernel.
يكلو لك (fork) process او يكلو لك (exit) terminate

* من من الآخر انت بتطلب من kernel بـ SysCall وهي تنفذ

E - Access to devices :-

اما تتيجي لتحكم الماوس او Keyboard او الماوس او Kernel فهو كلاته هو الـ Kernel و هو الذي يتمتع بـ Access

F- Networking :-

في المسؤولية عن بعثت الـ data على الـ network على الـ data الـ

G- Provision of sysCall APIs &

في الـ APIs التي يستخدمها بـ **برمجيات** (softwares).

H - Abstraction of virtual private computers

انت ال PC بتعالج ممكن يستخدمه أكثر من
user متاح ، المقادير هي تكونش سخال (in parallel)
الحقيقة لكن بين user لهم سخال (In parallel)
طبعاً اي ال Kernel بتوعنال الحوار

User mode :-

* الـ process يكون لها جزء مُخزن لـ memory في الـ RAM في بس الـ memory access و في الـ access run في الـ user mode الـ memory يدخل في الـ process الـ memory المطلوب

Kernel mode -

* رهال Kernel mode الى الى شکالاتي عنده بديل
الحق انك تتصفح كل الـ RAM وتحصل كل الـ Access
حتى لو في حاجات عندهم مُحرمة دللياً زع الى
الـ processor الذي يتوقف

Process View Vs Kernel View:

الـ *
الـ process فاكرة نفسها اذ لها كل access
كلـ الـ real RAM لكن في الحقيقة هي لها
على جزء منها بس و بيكوتش (لها access
على CPU مثل

*
كـ مـكن يـحـصلـهاـ حـاجـةـ (منـ كـامـلـةـ حـسـابـهاـ
لـهاـ) زـعـ اـنـهاـ يـحـصلـهاـ Scheduling وـ يـتـفـعـ مـكـانـهاـ
ـ تـاـيـرـةـ process

*
كـ مـشـ (فيـ) مـشـ مـكـانـهاـ فيـ الـ Memoryـ (منـ) ولاـ
هـتـشـتـكلـ اـمـتـ ولاـ هـتـقـفـ اـمـتـ ولاـ اـيـ حـاجـةـ
ـ كـسـ الـ Kernelـ الـ مـكـانـهاـ كلـ دـهـ

* The Shell *

* الـ Shell هي عبارة عن مترجم يترجم الـ User Commands إلى الـ Commands التي يفهمها المدخلات، وترجع لتنفيذ برمجيات تحمل الـ user حايلزه ويتـ execute في Command mode، فتـ قدر اـ تقول لها user mode interpreter

login Shell:-

هي الـ shell التي يـ مـ يـ كلـ user يـ مـ يـ كلـ /etc/Passwd بـ تـ خـ مـ الـ shell، فـ تـ لـ قـ يـ هـ اـ فـ يـ

Shell Versions:-

* أول Shell موجودة هي Shell اـ تـ خـ مـ الـ shell في التاريخ، اـ كـ يـ لـ لـ اـ فـ يـ كـ يـ شـ يـ

A- Bourne Shell (sh):

أـ كـ يـ شـ يـ قـ دـ يـ مـ مـ يـ شـ يـ، كـ يـ حـ يـ هـ اـ فـ يـ I/O redirection و globbing (C, *, .) و حـ يـ تـ اـ بـ يـ اـ فـ يـ، و فـ تـ لـ قـ يـ هـ اـ فـ يـ مـ عـ لـ مـ الـ Linux based OS بعد كـ يـ

B - C shell (csh) 8-

C-Korn Shell (Ksh)

لارج كورنر واحداً واحداً وخلافه ~~لارج كورنر~~
backward compatible مع CShell ثم مع bourne [L.E]

D-Bourne again shell (bash)%-

* دی کمپانیا بتّع GNO کلریفیتے علی Open Source بس bourneshell

Notes:-

* الـ sh موجودة في الـ simulation كـ bash . الـ sh يتبعها بـ features . الـ sh هو جملة بـ syntaxها يكتفى بـ روشتة . الـ sh هو جملة بـ syntaxها يكتفى بـ روشتة . الـ sh هو جملة بـ syntaxها يكتفى بـ روشتة .

Shell Scripts:-

* ده الحوار اللي ادناه ~~يحاكي~~ كام سين

يكتب في ~~file~~ run file في commands

و ممكن يبق فيه if, loop

Users and Groups



* كل User موجود كذلك في ال System كـ Unique user ID و Username موجود في /etc/passwd كـ Home directory و login shell كما مـ حـاجـةـ كـ دـهـ :-*

⇒ Group ID - Home directory - login Shell ←

* بـسـ الـ Passwordـ مـيـقـاشـ موجودـ فـتـالـيـ،ـ بـقـيـ موجودـ فيـ /etc/shadowـ وـتـقـرـرـ لـفـتـحـهـ وـاـنـتـ security wise كـ دـهـ لـ كـ دـهـ بـسـ!ـ يـمـ كـهـ لـ دـهـ



* الـ usersـ يـتـحـظـ وـاـمـعـ بـهـيـنـ فـيـ groupsـ الـ groupsـ الـ usersـ كانـ يـقـرـرـ يـخـلـ groupـ وـاـنـ بـسـ،ـ لـكـنـ BSDـ كـ خـلـتـهمـ يـخـلـواـ اـكـتـرـ مـنـ groupـ وـوـكـلـوـمـاتـ الـ groupـ موجودـةـ فـيـ /etc/groupـ وـفـيـ group name - group ID - user list



* الـ rootـ يـعـلـمـ كـلـ حاجـةـ فـيـ الـ Systemـ

* Single directory Hierarchy -- *

* الـ Single directory هيكل في windows أو Linux وهو الـ root والباقي تحته روابط
الـ link التي في صفحة `fd` كده بعدها مساحة `↑` مكتوب ارضاها مساحة `↑`

File Types:-

- 1- devices
- 2- pipes
- 3- sockets
- 4- directories
- 5- symbolic links «ShortCuts»

Directory:-

هيارة `table` هي `files` الـ `files` التي جواه
والـ `IO` بـ `reference` (او

Symbolic link:-
والـ `IO` بـ `target`
وـ `target` `file` تـ `target` `file` `target`
ـ `target` `file` `target` `file` `target`
ـ `target` `file` `target` `file` `target`

ـ `Symbolic` `Symbolic` `Symbolic` `Symbolic` `Symbolic`
ـ `Symbolic` `Symbolic` `Symbolic` `Symbolic` `Symbolic`

* لو في target اتنال symbolic link اللي كان يشير عليه بقائه dangling link

File names:-

* اسماء files اخرها 255 characters، ومهن تكون فنوا كل حاجة ماعاها null (\0) وال(/) slashes

* بس يعطل ان لا تستخدمن اعماles special characters من الاول لأن ال shell (مهن) تترجمها لحاجة تانية، ولو استخدمن خط (\) backslash، فاخذت نقط - او . او - بس اباقي هييف حوارات بالسالك

Pathnames:-

* الـ paths اللي عندنا فما نويين

* دو الـ absolute path > root المكان > cis path (Root) اللي انا واقف عنده ودالها بـ / (Root)

* دو الـ relative path > cis path (Root) اللي انا واقف عنده ودالها معينش / في اولها

* الـ base part و directory part cis path الـ directory او file او الـ base part اللي عايز اوجهه، الـ directory part دو من الاول لحد آخر حاجة عايز او حملها و مفديها، يكون كـ directories

;/;/;/;/home/ezzat/Desktop/main.c
directory Part ↪ basic Part

Current working directory:

* له المكان الى الى execute process ينبع مني مني و لوال Process ينبع مني وان يوجد له child process child process الى child process يكون له انتص الى cwd الى الى Parent cwd الى Parent موجود في /etc/passwd ويختبر SysCall ذاتي

File ownership & permissions:

* owner ينبع مني user ID ينبع مني Fileds كل ينبع مني owner group ID موجود فيه owner وال group ID موجود فيه

* كل File ينبع مني تبعيته لل users فهو ينبع من users لـ 3 حجات :-

A-Owner & File

ـ دخل اى هناك الى group owner ينبع مني

ـ اى دخل مني موجود في الـ 3 حجات الى group owner

* كل File ينبع مني 3 modes و قيم Access

A- read

B- write

C- search (directory) or execute (file)

File I/O Model

as a file بيتخاصل مع كل حاجة Linux *
 write(), read(), open() با تذریس SysCalls ک.

* و في هنالك lseek() وري بتخليك
 تقدر تتحروا من index من file معيشه بدل
 ماتت lseek() white ببردو، فهو بيعرك كل
 offset بدل ما يشاور على اول ال file بورده lseek() .

Parameters

Fd → الى File descriptor

offset → الى قيد القراءة من عند 0

whence → دى ليها 3 حيم

SEEK_SET:

cio offset bytes بعد pointer بخط

Pointer = offset file الى

SEEK_CUR:

cio offset bytes بعد pointer بخط
 مكان الحال

$$\text{Pointer}_{\text{new}} = \text{Pointer}_{\text{old}} + \text{offset}$$

SEEK-END:

دوى يتحطى الـ pointer على بىك offset bytes من نهاية الـ file يخلى الوالـ file كان آخره 20 - offset 1000 bytes ينكل بيه ايه فنكيف فنار

File descriptors

فهي باردة عن رقم يرجع من الـ sysCall بـ امتيازات open ويتستخدمه عـ شـ انـ تـ حـ اـ مـ لـ عـ بـ الـ file بعد كده ممكن او ده معلم الوقت، يكعـنـ رقم كـ بـ يـ

كارـة الـ 3 files بتـ inherits حـ والـ Process الـ Shell تـ بدأـ بـ

0 → STD_I/P

1 → STD_O/P

2 → STD_Error

الـ fd دول معلمـ الوقت متـ وصلـين بالـ terminal بـ سـ مـ كـ بـ يـ مـ تـ خـ يـ رـ حـ مـ زـ عـ ماـ كـ حـ لـ نـ اـ قـ بـ لـ كـ دـ

* Programs *

* اسکرپٹ کیا ہے؟ text is the script الی یہ کوں کہا رہے ہے؟
یعنی کان Shell وال Commands تھوڑے لوگاں
interpretor میں وی Python اور Shell Script

Ritters

Filter

* دی برامج بتخل ~~src code~~ کی ال ~~grep~~ کی ال
الی داخل زع ~~grep~~ وال ~~cat~~

Command Line Arguments :-

* دى الـ Command Arg. بالـ ~~Prog~~ الى يتبعها main Line \$ argv و args لازم تخلو الـ main خارج و

processes

Process Memory layout :-

- A - Text
 - B - Data
 - C - Heap
 - D - Stack

افهم بس الفرق بين الـ preemptive و الـ non preemptive

* RTOS Scheduler *

الـ scheduler هو المسؤول عن تحديد اتفى task الى شغالته وانتهى task للد، ويعتمد الوقت الـ scheduling على حسب الـ priority وكل task يتمسك بالـ CPU لحدة معيّنة تباعك تباعك تباعك

Non preemptive scheduling

الـ task الذي يتمسك بالـ CPU ومتخل في الـ running state يستمر في العمل منها إلا في ٣ حالات، أول حالة أوقفت كلما توقفت task، ثانية حالة أنها قررت تخرج من نفسها، الثالثة حالة اهتمامات، الـ MC حملها أو باطء مثلاً

طلب لوفي اي task اي priority منها جبت؟ تذكرة اما اخلعها ابقى ابيه الـ CPU، انا الـ CPU ماسكة الـ CPU وانا هما حية القرار الفوري في اني اسيب الـ CPU محدث يتحكم هنا

الحوار ده ليه كذا صيّدة، اه الـ Alg، اه الـ Alg، ببلايك Simple، اه الـ Alg، كذا high priority task عارف الـ task هستدياً وتخليها اهقي، وكمان تقدر high priority task عارفها، يعني انه الـ non-reentrant function ستحذف الـ SJF - FCFS ببلايك الـ low priority task

Preemptive scheduling:-

* ده الـ Preemptive scheduling هو higher priority (الهامي)، اللي لو حد بيتركه في وين، والـ CPU لو حد بيديه، بيتركه في على جنبه ويسكب higher priority، و ده الدكشن ثورة علما

* كده الـ Complex algorithm كده، كان الـ Preemptive scheduling

Examples:-

Round-Robin, Fixed priority Based scheduling

Look @ Slides 38 → 42