Session5 / /

* لو عندك proc1 و proc2 و proc3 ، و proc2 هي الـ parent بتاع proc3 ، و proc1 هي الـ parent بتاع proc2 ، وحبيت حملها termination يعني اتقفلت ، ساعتها الـ proc3 الـ parent بتاعها هيكون الـ init proc اللي هي الـ systemd و proc3 هيكون اسمها oriphan process يعني يتيمة

* بتقدر بالـ wait system call كـ parent (يعني هنا proc1) تعرف الـ child بتاعك مات ازاي (proc2) عن طريق ان في حاجة اسمها الـ task struct الـ kernel بتسيبه وتقدر تقرأ الـ state بالـ wait sysCall ، ولو الـ parent معملش read لـ state ، ساعتها الـ child بيكون في state اسمها zombie state مستنية القراءة

* و لو proc1 حملها termination ساعتها الـ state بتروح لـ systemd وهو بيتصرف فيها و تتمسح لانها واخدة مساحة عندك

* الـ child ممكن يموت لو خلص يعني خلصت البرنامج ، او انت قتلته بـ exit ، و في الحقيقة انك اما بتخلص الـ main يعني بتخلص البرنامج بترجع لـ start و هي بتنده الـ exit

* الـ status بتتخزن في شكل رقم integer و كل bit فيه بتدل على حاجة ، و في عندك حبة macros اقرأهم بتديهم الـ status و يقولك بيدل على ايه

echo $?   اللى فى اخر الـ main

* بيجيبلك الـ ~~[crossed out]~~ return بتاع اخر process كان شغال

* اول 8-bits من الـ integer بتاع الـ status بيعرفلك الـ return بتاع الـ process ، بس خلى بالك ان الوضع ده ممكن يتغير عادى فا ستخدم الـ macros

* الـ bash عندها scape char زى الـ (\ ) → backslash او انك تحط اللى انت عايزه بين " " او ' '

* touch * → All Files

* touch *hello → All files ends with hello

* touch ?hello → All file have 1char then hello

* touch [12]hello → ~~All~~ 1hello or 2hello

* touch [1-9] hello → 1hello, 2hello → 9hello

* touch [1-99] hello → 1hello, 2hello → 9hello & 9hello

* الـ bash ممكن تخزن فيها variable عادى و بيتخزن فيها بشكل string ، x = value ، تقدر تعرضه بـ echo $X ولو عايز تمسحه unset variable ، و لو عايز تعرض اسماء الـ variables كلها بقيمهم استخدم set بدون arguments

/ /

* الـVariable اللى احنا عملناه د Variable local للـ process دى ، لو عايزه يبقى ظاهر للباقى يبقى استخدم export واسم الـvariable بعد ما تعمله local

export X → enviroment Variable

* لو عايز تعرض الـ.enviroment Var كلها استخدم env

* لو عايز تعمل enviroment Var مش لازم يكون موجود ممكن تستخدم export على طول

export Y= 10

* لو عايز تمسح الـEnv. variable اعمله unset

* الـenv. var دى بتتبعت للـmain كـthird arguments كمان الـexecve الـthird arguments اللى قولنالك هسيبه بعدين

* فى عندك env. variable اسمه PATH ، ده فيه الـPATHES بتاعت الـprograms اللى مش built-in فى الـbash فلما تيجى تعمل execute بيروح يدور فيها.

* ممكن تزود فيه path عن طريق PATH = $PATH + ":..."

* لو عايز ت run عندك bash script من File ، روح حط فيه الـCommands بتاعتك ، وغير الـmode بتاعه لـexecutable بـChange mod

chmod +X Filename

* واعمله run هيـexcute الـCommands كلها اللى فيه

/ /

* لو انت حابب تـ run مثلا Python script لازم تديله الـ path بتاع الـ interpreter اللى هتستخدمه فى اول الـ file

```
#!~Path_of_interpreter "Python"
printC "Hello")
```
(no space)

* لو انت محطتش #!. الـ default بتاعك هو الـ bash.

* لو انت عرفت local var جوا الـ script مثلا و عملت run زى ما احنا عملنا، هتلاقيه مش موجود فى الـ bash الاساسية

* ممكن تستخدم source، و ده هيـ run الـ file فى الـ bash اللى مفتوحة حاليًا، و ده هيخلى الـ var، local ليها فهو جود عادى

* الـ enviroment var. بتتبعت من الـ parent لـ الـ child، و ممكن منبعتش كده اساسا، و لو الـ child عدل الـ parent مش هيحس بالتعديل ده لانه بيحمل فى الـ memory بتاعت الـ child بس، بردو لو زودت enviromentVariable فى الـ child الـ Parent مش هيحس بيه

/ /

* ممكن تجيب الـ environment var باستخدام environ ، ده global var بتعمله extern وتستخدمه بدل ما تبعت للـ main الـenvironment variables كـ third Parameter

* الفرق بين الـ " " والـ ' ' ، إن الـ " " بـ expand الـ variables اللى فيها ، اما الـ ' ' بيحولا يعنى لو عندك X=5 و عملت echo "$x" هيطبع 5. اما echo '$x' هيطبع $X .

echo $$
↳ print PID of Current bash

proc:-

* ده filesystem مش حقيقى الـ Kernel عمله فى الـ RAM يعنى ، الـ Kernel بتستخدمه عشان تقول معلومات informations للـ user عن الـ processes اللى شغالة على شكل pesudo-file

~~ls/Proc~~ ls /proc

* هتلاقى فيه شوية ارقام ، دول الـ ID بتوع الـ processes اللى شغالة ، و حبة حاجات عن الـ uptime والـ cpu فى files ، ومعظم الـ files دى readonly بس فيه writable عادى

* مع كل proc بتتعمل الـ Kernel بتعملها file واما الـ proc بتتقفل ، الـ file بيتمسح

/ /

* كل Folder لـ process جوا /proc، بيكون جواه files، كل File شايل معلومات عن الـprocess حبة تقدر تعمل man 5 proc و تشوف كل واحد فيه ايه

* من ضمن الحاجات اللي موجودة # جوا كل process Folder هي fd ودي بتقولك الـfiles اللي الـprocess مستخدماها و ده مهم هنستخدمه كمان حبة

* في عندك كمان cmdline بيقولك الـprocess دي مين كملها

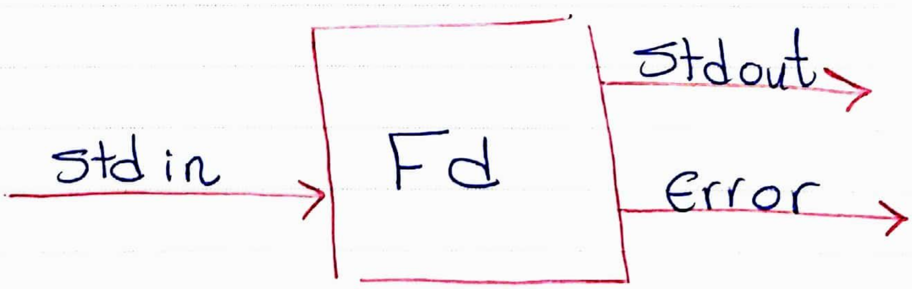
* كمان عندك environ بيقولك الـenviroment Variables اللي ابتعتت لـexecve وهي بتكمل الـprocess دي

How IO redirection is done?

انت لو روحت على اى Folder بتاع Process،
و كملت لـ ls fd انك تشوف الـ File descriptor بس
فى الـ long format، هتلاقى Files ليها ID 0 و
1 و 2 وحبة Files تانية

* 0 → input file «STD in» Keyboard by default
* 1 → output file «STD out» screen by default
* 2 → error file

* بص انت كل process عندك بيكون ليها i/ps
و o/ps فى الـ File description

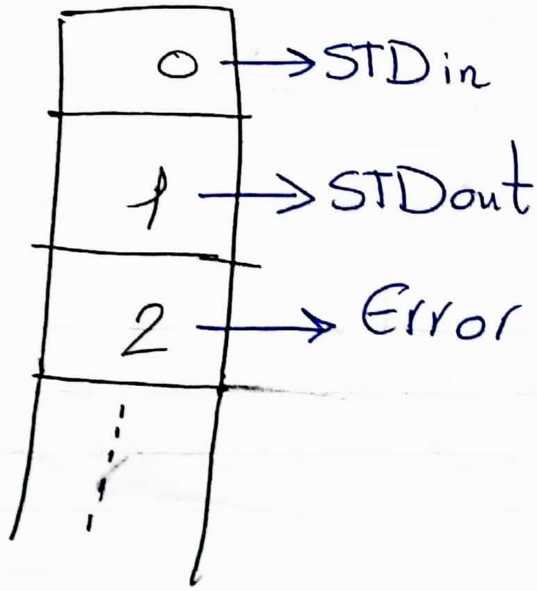


* الـ STDin دى عبارة عن الـ i/p اللى داخل الـ system
و by default بيكون الـ Keyboard ، بس انت
تقدر تغيره عادى بالـ i/p red. اللى عملناه قبل كده
وتخليه ياخده من file

* الـ STDout دى عبارة عن الـ o/p اللى بيطلع من
الـ process ومعظم الوقت بيكون الـ screen
by default وممكن انت تغيره عادى زى
الـ o/p redirection اللى عملناه قبل كده

* الـ Error دى رسالة الـ error اللى بتخرجلك
على الـ screen لو فى مشكلة وتقدر تعملها
redirection عن طريق 2> fileName كمثال ls 2>Error.txt
كده لو فى error هيطلعلك على الـ screen هيتحول
لـ Error.txt

* طب الـ Files دى بتتخزن فين ، بيكون عندك
data structure اسمها per process file desc. table او
اختصارًا PPFDT بتكون بالمنظر ده



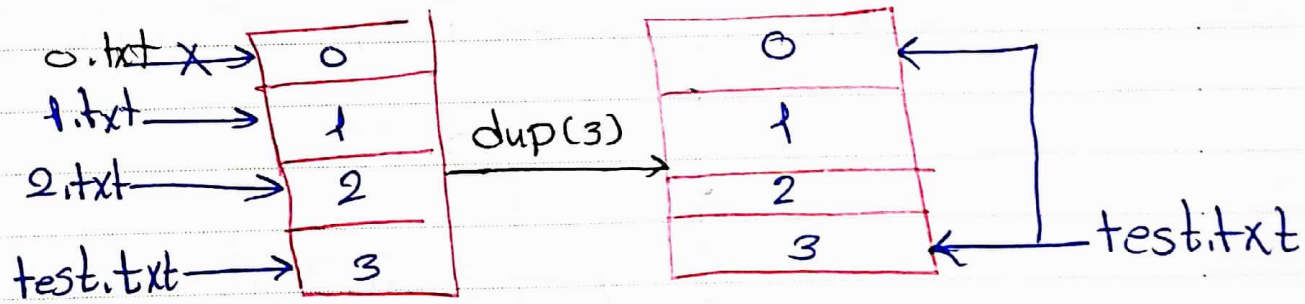
Note:-

* الـ STDout والـ Error ممكن يبقى الـ file متن موجود هيكمله عادي، اما الـ STDin لازم يبقى موجود اجباري، هيقرأ منين؟ 😂

dup & dup2 SysCall:-

* دلوقتي احنا عرفنا ازاي تقدر تعمل redirection، بس ده عالـ shell او الـ bash، طب انت لو عايز تبعت الـ o/p او الـ error او تاخد الـ i/p من file غير كده عالـ sys بتاعك by default متن الـ user اللي يعمل كده هتعمل ايه؟

* في عندك SysCall اسمها dup، بتديها الـ Fd القديم بتاعك، وهي بتدور على اول fd موجود عندك وتـ link له بيه، بالشكل ده:- وفاضي



* كده انت الـ test.txt كان linked بـ 3، و اما عملت dup(3) راح دوز على اقل fd وعمله link بيه بردو اللي هو الـ (0)، طب حلو، فعمل ايه بردو؟

/ /

1] open test.txt *
اول حاجة هتروح تفتح الملف اللى انت عايز
تحطه كـ STDin/p كشان تتأكد انه شغال

2] get FD of test.txt
تجيب الـ fd بتاع test.txt اللى انت هتحطه
كـ STDin/p

3] close o.txt:
تقفل o.txt اللى انت عايز تحط مكانه test.txt

4] dup(Fd_of_test.txt)
تروح تحط بقى test.txt فى zero

فى الحتة دى ممكن فى file يحصله dup، بالتالى
dup2 مش هيتحط كـ STDin، فكشان كده استخدم dup2
بتاخد الـ fd القديم والجديد

```
dup2(4, 0);
```

ALADDIN