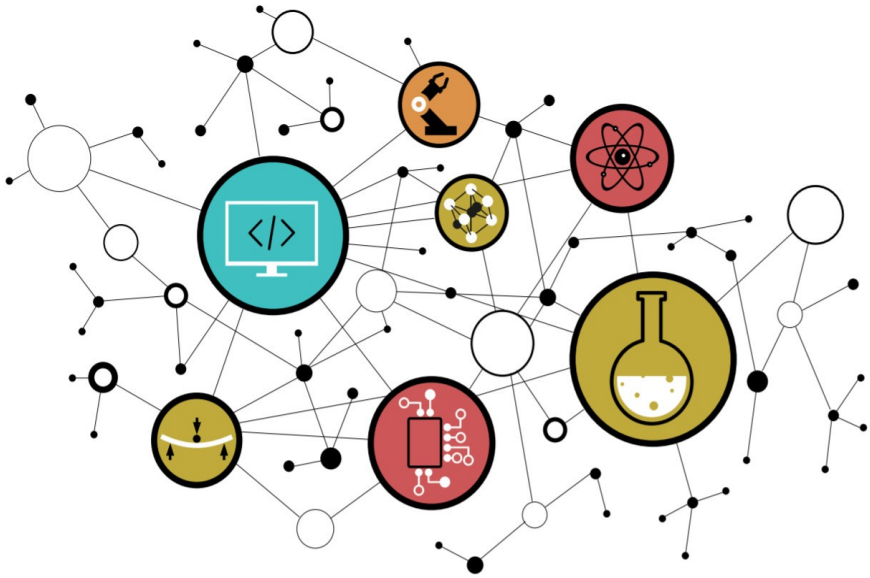
TRAVAUX PRATIQUES

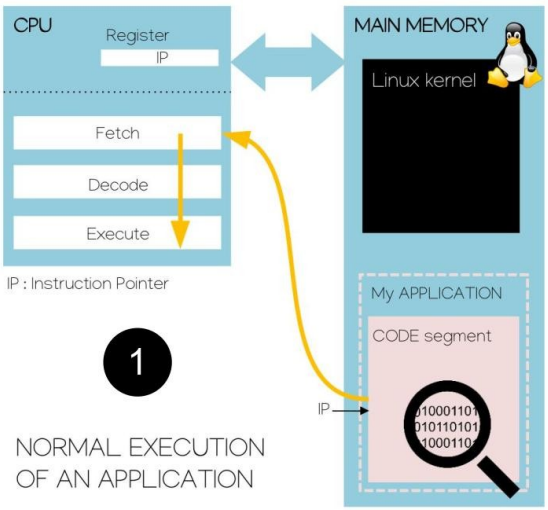
EXCEPTIONS MATÉRIELLES ET SIGNAUX UNIX

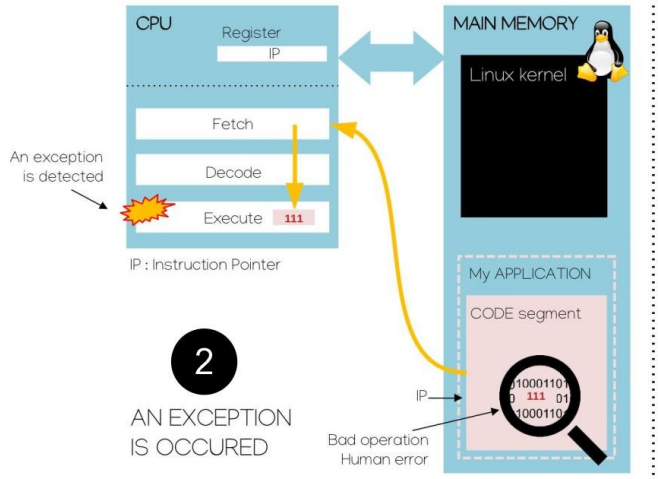


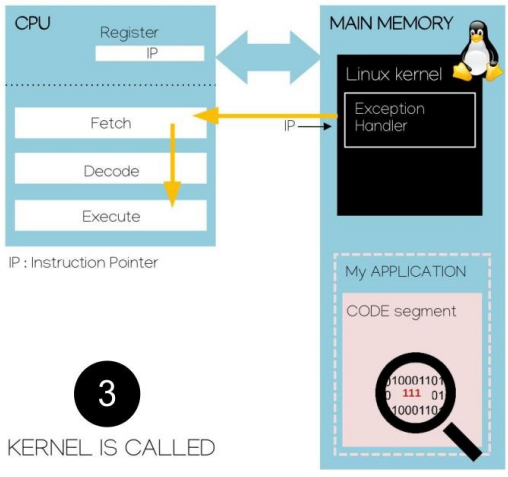
Hugo Descoubes – neku – GNU\Linux 5.8 - Ubuntu 20.04 LTS – LibreOffice 6.4.6.2 - 2021

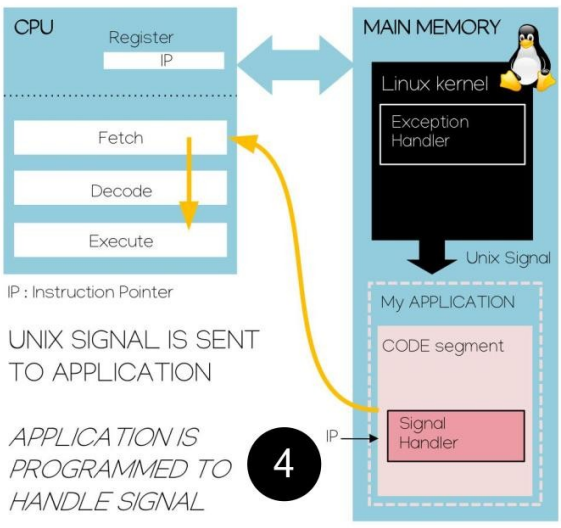
SOMMAIRE

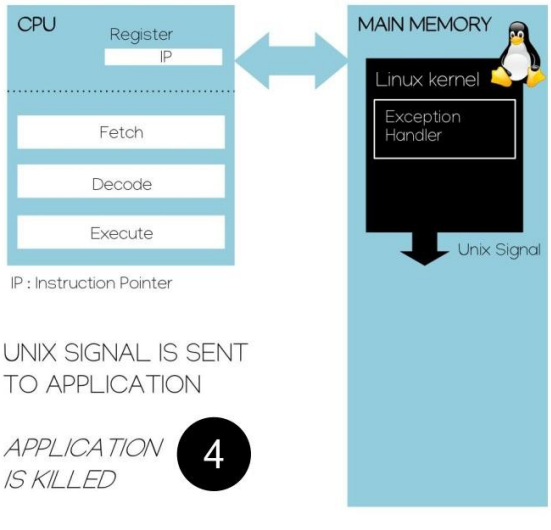
7. EXCEPTIONS MATÉRIELLES ET SIGNAUX UNIX











Une exception matérielle est un événement matériel synchrone généré par le CPU voire la MMU. Nous parlons d'événement synchrone au regard du fonctionnement d'un CPU dont les traitements restent synchronisés sur une référence d'horloge et non au regard de la probabilité d'occurrence. Une exception peut interrompre l'exécution d'une application à tout moment. Ces événements sont relevés par le CPU lorsque celui-ci détecte une condition prédéfinie et documentée non conventionnelle faisant exception durant l'exécution d'une instruction (violation de privilège, division flottante par zéro, accès illégal en mémoire, etc). Contrairement aux *interruptions* (générées par les périphériques) provoquées par des causes externes au programme (asynchrone), les *exceptions* sont provoquées par des causes internes au programme (synchrone). Toute exception est une opération connue ne devant généralement en aucun cas arrivé. Elles sont le plus souvent le fruit d'une erreur de programmation. Détaillons la séquence graphique présentée sur la page précédente :

1. L'application s'exécute normalement sur le CPU courant. Le noyau est au repos en attente d'un événement requérant son travail

2. Le programme en cours d'exécution implémente une opération binaire erronée. Durant son exécution par le CPU, l'instruction génère une exception au fonctionnement normal du CPU. Le pointeur d'instruction IP est alors redirigé (par lecture et exécution d'un tableau de vecteurs) vers une fonction noyau dédié au traitement des exceptions. Le noyau du système s'exécute alors sur le CPU courant

3. Le CPU vient donc de stopper l'exécution de l'application en cours et d'appeler une procédure enfouie du système. Une sauvegarde du contexte CPU (registres de travail internes) est également réalisée et pourra être accessible depuis l'application en espace utilisateur. La procédure de gestion des exceptions est implémentée par la fonction *do_page_fault* dans le cas de Linux (présente dans le fichier */arch/<cpu_architecture>/mm/fault.c* du système de fichier du kernel Linux, https://www.kernel.org/). La fonction de gestion des exceptions (ou *exception handler*) est chargée de relever le type de défaut et de générer, si l'exception le permet, un signal logiciel Unix à destination du processus (application) ayant généré le défaut.

4. (schéma de droite) Si le processus ne traite pas le signal Unix, il est alors tué par le noyau du système qui assure la libération de toutes les ressources mémoire associées. Aucune autre application n'est impactée. Ceci assure un cloisonnement des défauts et la stabilité du système.

   ou

   (schéma de gauche) Si le processus traite le signal Unix (code spécifique intégré à l'application), l'application peut alors tenter d'acquitter le défaut (restaurer un contexte CPU viable) ou tenter une solution de contournement (redirection vers un autre code de l'application viable, redémarrage ou mode dégradé de l'application, etc). Si cela est possible, un message d'erreur ou un fichier de log pourra être sauvé voire envoyé aux équipes de développement.

7.1. Lecture seule

- Se placer dans le répertoire *disco/except*. Compiler le fichier *except_readonly.c* jusqu'à l'édition des liens incluse et exécuter le programme. Interpréter et expliquer l'exception matérielle ainsi que le défaut logiciel à la racine de cette exception. Proposer un schéma commenté.

```
gcc except_readonly.c -o except_readonly

./except_readonly

Segmentation fault (core dumped)
```

7.2. Pointeur nul

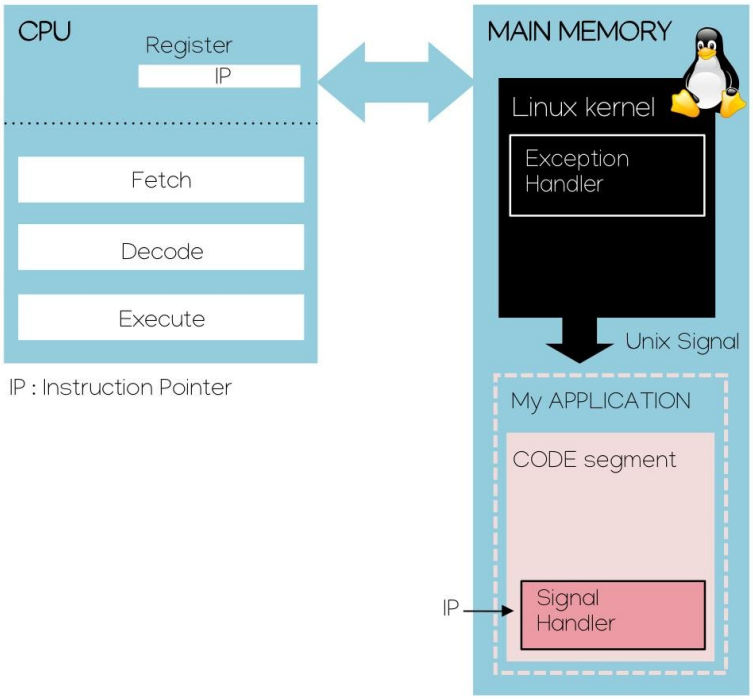
- Compiler le fichier *except_null_pointer.c* jusqu'à l'édition des liens incluse et exécuter le programme. Interpréter et expliquer l'exception matérielle ainsi que le défaut logiciel à la racine de cette exception. Proposer un schéma commenté.

```
gcc except_null_pointer.c -o except_null_pointer

./except_null_pointer

Segmentation fault (core dumped)
```

7.3. Signal Unix



- Compiler le fichier *signal_handler.c* jusqu'à l'édition des liens incluse et exécuter le programme. Interpréter et expliquer l'exception matérielle ainsi que le défaut logiciel à la racine de cette exception. Analyser le script assembleur du programme et préciser l'instruction ainsi que le registre à la source de l'exception. *La séquence qui suit propose de remplacer le contenu du registre avec une valeur acceptable par le système pour la bonne exécution du processus.*

```
gcc signal_handler.c -o signal_handler

gcc -S -fno-asynchronous-unwind-tables -fno-pie -fno-stack-
protector -fcf-protection=none -Wall signal_handler.c
```

- Dé-commenter la section de code assurant la configuration et la gestion des signaux système reçus par l'application. Analyser le fonctionnement du programme.

- Dé-commenter la ligne de code dans la fonction *signal_handler* proposant une solution alternative afin d'acquitter l'instruction à la source de l'exception (dans le *main*) et analyser le résultat. *Voilà, vous venez de générer volontairement une exception (défaut de segment), de capter le signal Unix envoyé par le système, d'acquitter l'erreur par remplacement du contenu d'un registre (RAX) et de redonner une chance à l'application de s'exécuter sans défaut.*


notes