INF720 - Langage C et Systèmes d'Exploitation

Processus 1ière partie

Responsable : Thomas Robert (C234-4)
Auteurs : Thomas Robert

Le modèle d'exécution d'un programme impératif – les appels et les paramètres ...

En C :

- Un appel de fonction
  - Copie la valeur des paramètres dans un espace dédié à l'appel en cours
  - Mémorise l'orgine de l'appel
  - Réserve de la place pour stocker le résultat et assurer sa manipulation une fois l'exécution terminée
- La structure de donnée associée : la pile
  - Pouquoi une pile ? Car on enchaine les appels et qu'il faut potentiellement conserver la valeur des paramètres (exemple au tableau)
  - Parce que les données les plus utiles sont les dernières stockées

Ex d'une illustration simplifiée ici

Capacités de traitement coté processeur

- **Composant clé : le coeur d'exécution**
- **Traitements = instructions (opérations *élémentaires*)**
- **Jeu d'instructions = {des opérations élémentaires supportées par le processeur}**
  - Stockées en mémoire, donc
    - représentation binaire
    - Chaque instruction élémentaire à une adresse
  - Jeu d'instruction limitées
    - Calculs numériques simple (+ ; - ; * ; / …)
    - Transferts de données entre supports de stockage (mémoire registre ou registre registre ou registre mémoire)
    - Evaluations de conditions booléennes
    - Utilisent principalement les registres (pour des questions de performance)
  - Si mécanisme de protection : notion d'instructions privilégiées vs non privilégiées

- **2 protections différentes HW :**
  - **contrôle des accès en lecture / écriture sur la mémoire**
  - **Contrôle des instructions exécutée (partitionnement du jeu d'insctruction en 2)**

Le cas particulier du Déroutement avec restauration

On veut calculer

$$\sum_{j=121}^{242} j$$

Rappel Somme 1…N = N*(N-1)/2
=> calcul possible en O(1)

idée N dans Ra et résultat dans Rb
Rb doit contenir ((Ra-1) * Ra)/2

Calcul de N*(N-1)/2 :

@	Instruction
911	Rb ← Ra-1
912	Rb ←Ra*Rb
913	Rb ←Rb/2
914	….
915	….

Le cas particulier du Déroutement avec restauration

On veut calculer

$$\sum_{j=121}^{242} j$$

Rappel Somme 1…N = N*(N-1)/2
=> calcul possible en O(1)

idée N dans Ra et résultat dans Rb
Rb doit contenir ((Ra-1) * Ra)/2

Calcul de N*(N-1)/2 :

@	Instruction
1	Ra ← 120
2	Call 911
3	Rc ← Rb
4	Ra ← 242
5	Call 911
6	Rb ← Rb -Rc

@	Instruction
911	Rb ← Ra-1
912	Rb ← Ra*Rb
913	Rb ← Rb/2
914	Ret
915	….

Le cas particulier du Déroutement avec restauration

Rappel Somme 1…N = N*(N-1)/2
=> calcul possible en O(1)

On veut calculer

242

Rb

Bilan :
Des registres pour les paramètres,
pour la valeur de retour, pour l'adresse où l'appel a été réalisé ….

Que fait on si il y a plus de donnée à passer en paramètre que de registres ?

=> il faut paser par la mémoire
Solution = concept de "stack frame" ou bloc de pile
1 structure placée en mémoire qui contient toutes ces données.
+
1 pile pour stockée plusieurs structure en fonction de la logique d'exécution

6	Rb ← Rb -Rc

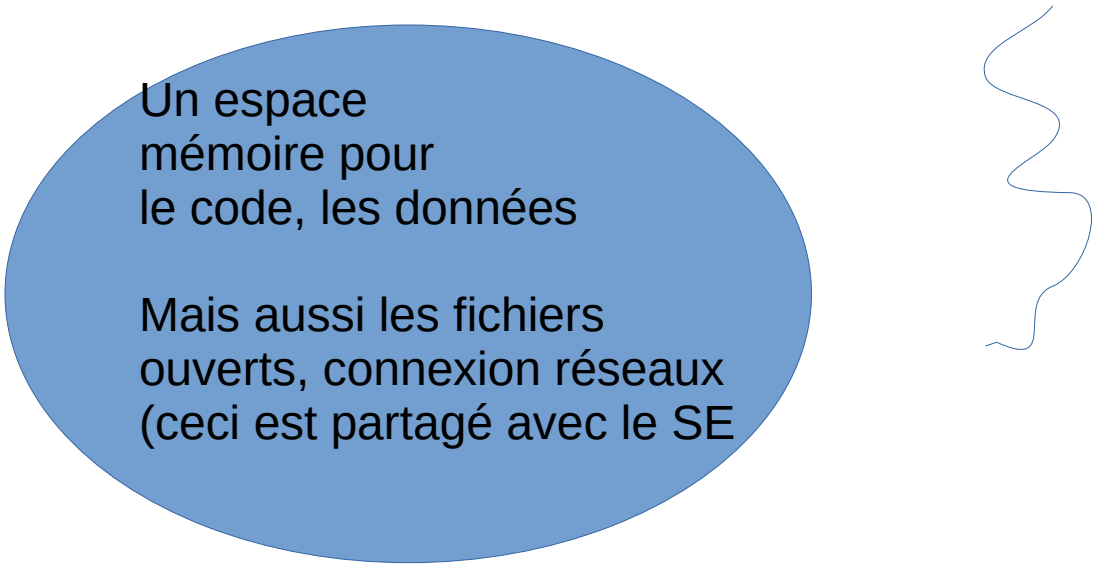
915	….

Etat d'exécution et mécanismes de protection

- **Idée de la logique de protection :**
  - **Majoritairement encodée via le contrôle d'accès à la mémoire, lui même encodé en mémoire sur une zone uniquement accessible par le système (« espace » noyau)**
  - **Pas d'accès direct au périphérique par un code applicatif (instructions privilégiées)**
  - **Instruction spéciale pour passer au mode privilégié garantissant l'absence de contournement des protections**

- **Etat de l'exécution d'une séquence d'instruction =**
  - **1 registre pour la prochaine instruction**
  - **1 registre pour l'adresse du sommet de pile (avant se situe les information de l'appel de fonction en cours, et on insérera après les informations de tout appel à venir.**
  - **N registre de « calcul » pour stocker de l'information**
  - **N registre d'état du processeur (la lecture/modification de ces registres peut recquerir des instructions privilégiées).**

Le concept de context d'exécution



Des structures de données pour gérer l'exécution d'une séquence d'insctructions

Pile,
Sauvegarde des registres en cas de suspension

Processus = l'espace mémoire + les structures pour gérer l'exécution d'une séquence au moins

Usuellement 4 scénarios en informatique

- 1 processus 1 séquence
- 1 processus N >1 séquences (partage de la mémoire) mais parallélisme
- N processus 1 séquence
- N processus N séquences

Pour la suite polycopié INF104 adapté à INF720

Appel système clone

Clone : appel système dit primitif pour la création de contexte d'exécution

- **Syntaxe : int clone(int (*fn)(void *), void *child_stack, int flags, void *arg)**
- **int (*fn)(void *) : adresse du point de départ du nouveau processus**
- **void *child_stack : espace mémoire pour la pile du nouveau processus**
- **Int flags : paramétrage de la création (cf page de manuel pour plus de détails …)**
- **Void* arg : paramètre utilisé lors de l'appel du point d'entré du processus.**

Appel système clone

L'intuition :

- **Le nouveau contexte d'exécution ne peut exécuter qu'une nouvelle fonction**
  (pas de poursuite de l'exécution de main comme avec fork)
- **Le nouveau contexte peut partager certaines ressources du processus appelant clone**
  - **Extrême 1 : rien du tout, ce cas est activé par le choix du "flag" 0**
  - **Extrême 2 : tout sauf la pile d'exécution, activé par une combinaison de constantes prédéfinie : C1 | C2 | C3 (chaque constante défini un ou plusieurs bits désignant ce qui est partagé ou pas à la création du contexte cf manuel)**
- **Il faut prévoir à l'avance un nouvel espace pour la pile d'exécution car même dans le cas 1 la pile de l'appelant même si elle est copiée ne peut être réutilisée !**