Le Chien de garde / Watchdog

- Uniquement dédié à garantir la fonctionnalité du système
- Durée d’attente fixe, à quelques options près (postscaler)

- Actions possibles
  - du watchdog sur le processeur (lorsque la durée est atteinte):
    - Réveil du processeur (si en sommeil)
    - Reset du système
  - Du processeur sur le watchdog :
    - Reinitialiser le temps d’attente
    - Démarrer le watchdog (mais impossible de l’arrêter !!)

Le Chien de garde / Watchdog

- Usages :
  - Immédiat : blocage des fenêtres windows (logiciel uniquement)
    - (cette fenêtre ne répond pas)
  - Equipements réseaux (téléphonique) ou satellites…
  - Informatique embarquée au sens large…
  - Cas particulier : transports...

Timer périodique

- Timer spécifique fait pour avoir une période précise et stable…
  - Impossibilité de gérer précisément la phase (synchroniser le timer sur d'autres évènements)

- Utilisé en TP car accès *facile*…
- 1 bit de statut :
  - passe à 1 en fin de période
  - remis à 0 par le processeur

Interruptions...

choix de la prochaine instruction éxécutée par le processeur

- Si une interruption est demandée
  - La prochaine instruction est à une adresse prédéterminée, quoi qu'il arrive (l'emplacement de l'instruction qui aurait du être la suivante est sauvegardé)
- Sinon
  - Si c'est un branchement
    - Éxécuter l'instruction indiquée par le branchement
  - Sinon
    - Si c'est un test
      - La prochaine instruction est la suivante, ou celle juste après, selon le résultat du test
    - Sinon
      - La prochaine instruction est la suivante

Interruptions

- Concept : Un processeur n'obéit pas QUE aux instructions de son programme…
  - Il répond également aux entrées d'interruption
  - Chaque ligne d'interruption est associée à l'adresse d'une fonction à éxécuter (vecteur d'interruption)
  - Le vecteur contient un branchement vers une fonction complète (configurable)
- Chaque périphérique parle sur un seul vecteur, mais plusieurs périphériques peuvent se partager un vecteur...
- Déclenchement *asynchrone* (d'un point de vue algorithmique)

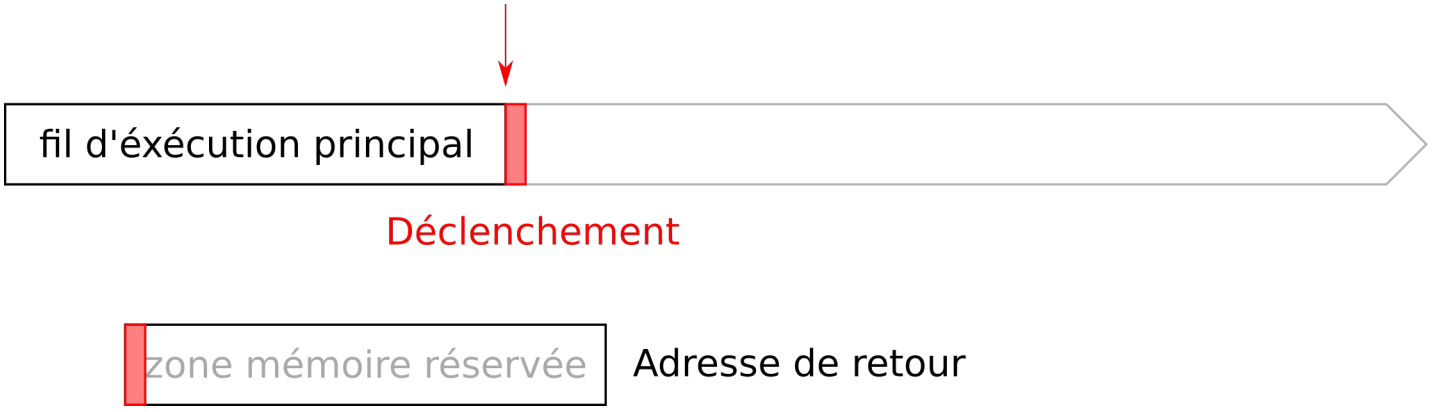
  => doit fonctionner sans perturber le fonctionnement de base.

Interruptions

- Déroulement temporel

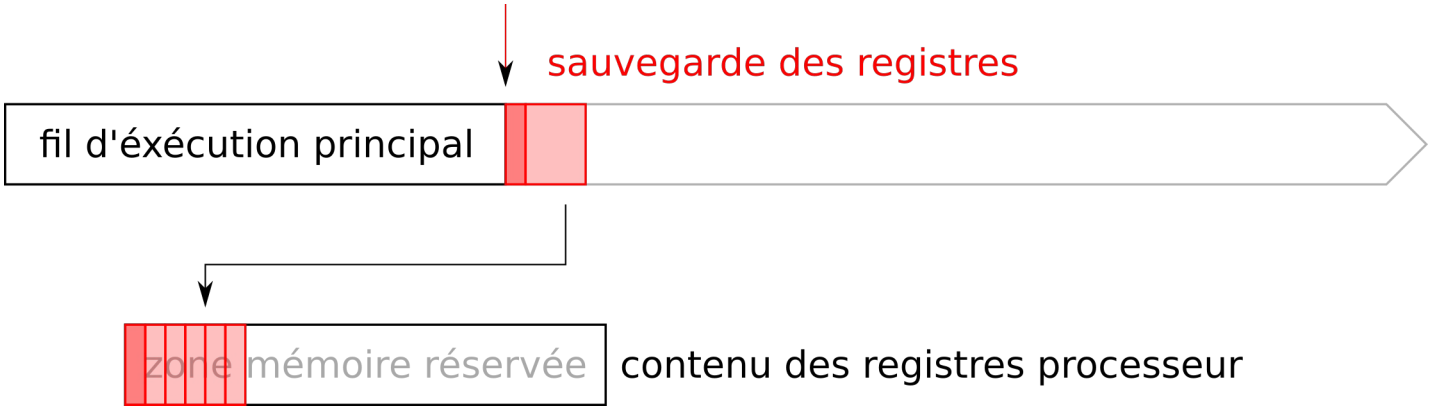
Interruptions

- Déroulement temporel



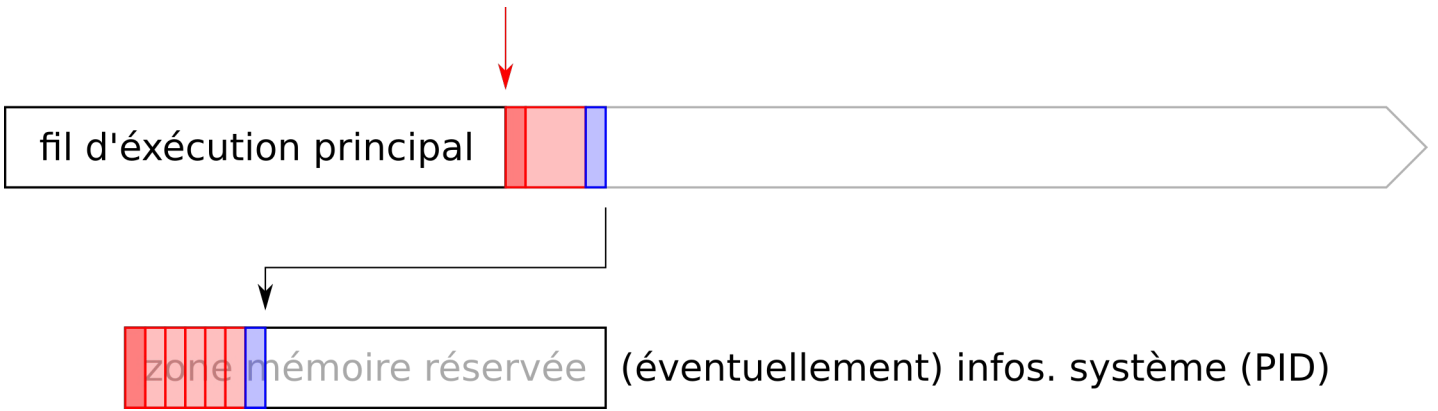
Interruptions

- Déroulement temporel



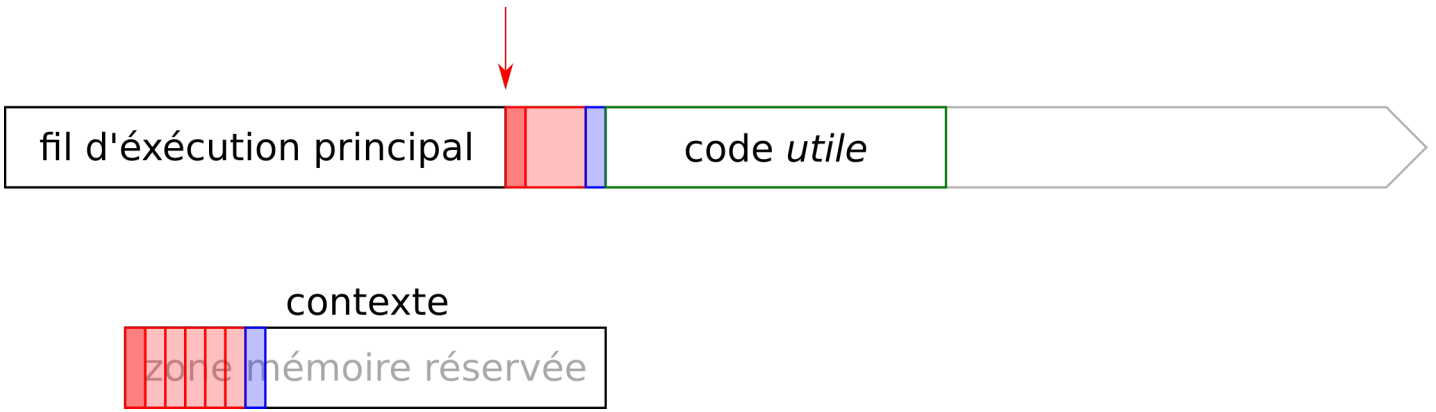
Interruptions

- Déroulement temporel



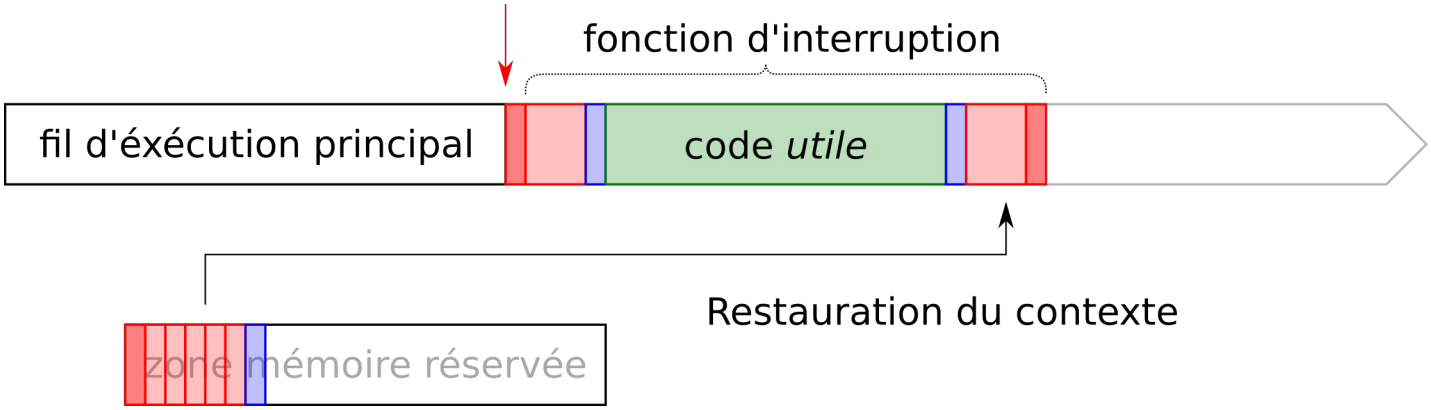
Interruptions

- Déroulement temporel



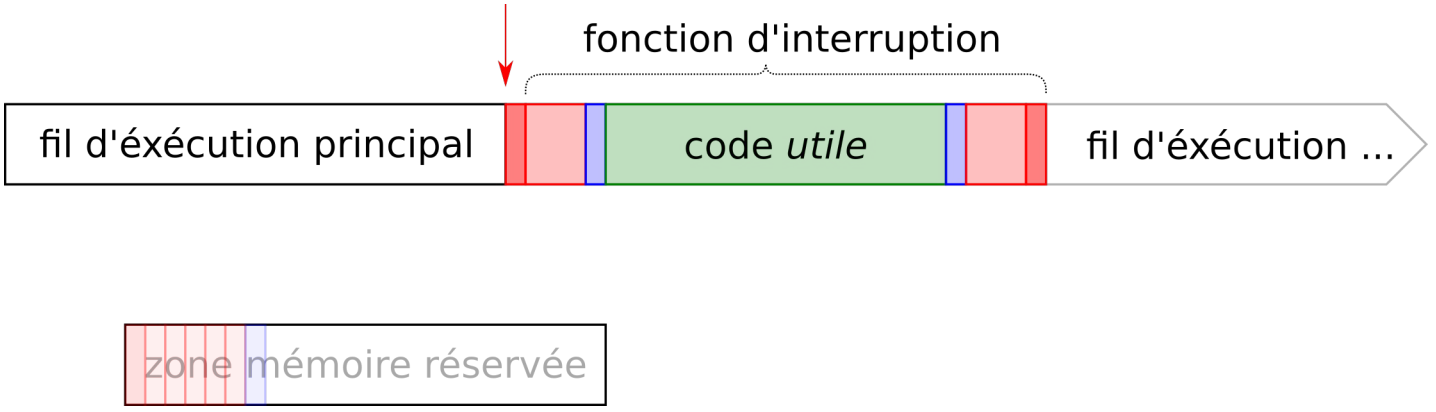
Interruptions

- Déroulement temporel



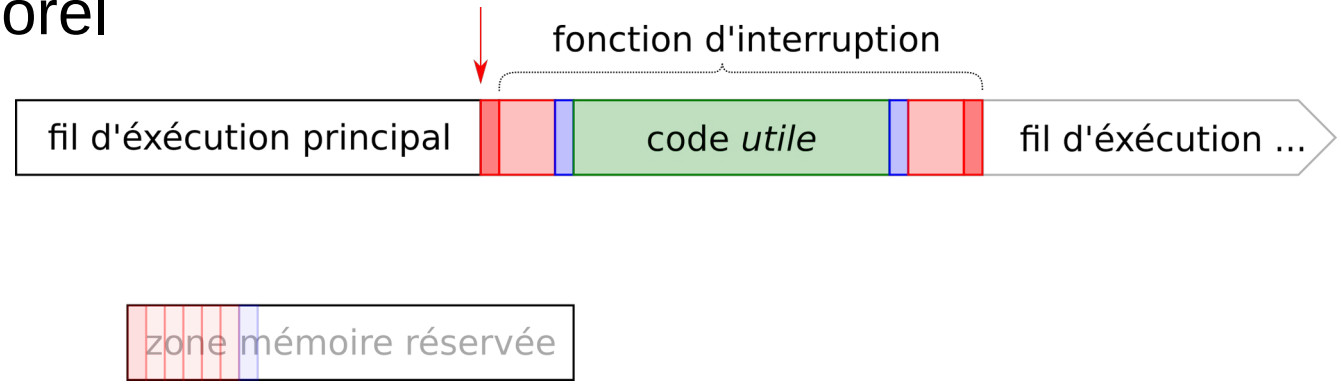
Interruptions

- Déroulement temporel



Interruptions

- Déroulement temporel



- L'état du processeur est rigoureusement identique avant et après l'interruption
- Première approche multitâche : les opérations du système s'effectuent *en arrière plan*

Utiliser les interruptions...

- Très spécifique, dépend de :
  - à l'environnement de compilation
  - au processeur cible

- Machines simples :
  - On ajoute le mot-clé *interrupt* devant la fonction…
    ex: `void interrupt fct_it(void)`
  - Le compilateur s'occupe
    - d'ajouter la sauvegarde et la restauration du contexte en début et fin de fonction
    - de placer le code de la fonction à l'endroit qui va bien (ou de faire un lien)

Utiliser les interruptions...

- Machines plus complexes :
  - Nécessite un sous-système de gestion (assembleur ?)
    - Sauvegarde/restauration du contexte
    - Réduction de droits éventuels
  - La fonction d’interruption est une sous-fonction
    - Fonctionnement classique d’une sous-fonction
    - Peut-être mise à jour / échangée à chaud
    - Possibilité d’utiliser la fonction sur plusieurs vecteurs