studio shodwe

OPTIMISATION DE CODE

VACHER Rose & BREUIL Clément

Start

INTRODUCTION À L'OPTIMISATION DE CODE

A

Le but de l'optimisation

- Reduction du temps d'écution
- Reduction de l'espace memoire
- Amelioration de la consomation d'energie
- Rationallisation des ressources

POURQUOI OPTIMISER LE CODE

- Code optimisé = meilleures performances.
- Réduction de l'utilisation de la mémoire.
- Réduction du temps de traitement.

TYPE D'OPTIMISATION

Trois types d'optimisation :

1. Optimisation des performances
   - Améliorer la vitesse et la réactivité

1. Optimisation de la mémoire
   - Réduire l'utilisation de la RAM et éviter les fuites de mémoire

1. Optimisation de la lisibilité et de la maintenabilité
   - Rendre le code plus compréhensible et facile à maintenir

APPROCHE D'OPTIMISATION

- Optimisation algorithmique : Choix d'algorithmes plus efficaces
- Optimisation du langage de développement : Ordonnancement optimal des instructions
- Optimisation avec un langage de bas niveau : C, assembleur pour un contrôle direct du matériel

OPTIMISATION DES ALGORITHMES

- Sélectionner des algorithmes avec une meilleure complexité
- Exemple : Remplacer un tri à bulle par un tri rapide

OPTIMISATION DE LA GESTION DE LA MÉMOIRE

1.

Gestion efficace de la mémoire

- Allocation et libération appropriées
- Éviter les fuites de mémoire

2.

Optimisation du cache

- Utilisation du cache pour améliorer les performances

OPTIMISATION DES ENTRÉES/SORTIES (E/S)

- Réduction des accès aux disques
- Mise en cache des données
- Optimisation des requêtes et flux de données

OPTIMISATION DU MULTITHREADING ET DE LA PARALLÉLISATION

- Utilisation des processeurs multicœurs pour exécuter plusieurs tâches simultanément
- Gestion des conflits d'accès à la mémoire partagée

OPTIMISATION POUR DES ENVIRONNEMENTS SPÉCIFIQUES

1.

Systèmes embarqués

Réduction de la consommation d'énergie et de mémoire

2.

Applications mobiles

Minimiser la consommation d'énergie

2.

Serveurs et cloud computing

Gestion de la scalabilité et de la répartition de charge

DÉFIS DE L'OPTIMISATION DU CODE

- Optimisation prématurée
- Compromis entre lisibilité et performance
- Portabilité du code
- Coût du temps de développement

OUTILS ET TECHNIQUES POUR MESURER LES PERFORMANCES

1.

Profilers

- Permettent d'analyser le temps d'exécution des différentes parties du programme et de détecter les goulots d'étranglement.

2.

Analyseurs de mémoire

- Aident à identifier les fuites de mémoire et à optimiser l'utilisation des ressources mémoire.

2.

Gestion des dépendances

- Permettent d’optimiser l’utilisation des bibliothèques et de s’assurer que seules les versions nécessaires sont utilisées.

CONCLUSION

- L'optimisation du code est essentielle pour améliorer l'efficacité des programmes.
- Doit être réalisée de manière réfléchie, en équilibrant performance et lisibilité.
- Utilisation des bonnes pratiques, outils de profilage et tests pour garantir des programmes performants et maintenables.

MERCI

Avez vous des questions ?