ECOLE SUPERIEURE D'INGENIEUR DE RENNES

Université de Rennes

SYS-S5 Structure d'un SE 3/12

Florian Cliquet

Note Taking of Online Resources

9 juin 2024

Plagiarism Mention

We attest that the content of this document is original and stems from our personal reflections.

Sommaire

Introduction

This document provides an overview of SYS-S5 concepts written by Cliquet Florian. It's a set of notes on multiple online resources.

1 Structure d'un SE

1.1 Interfaces utilisateur

Les Types d'Interfaces utilisateur :

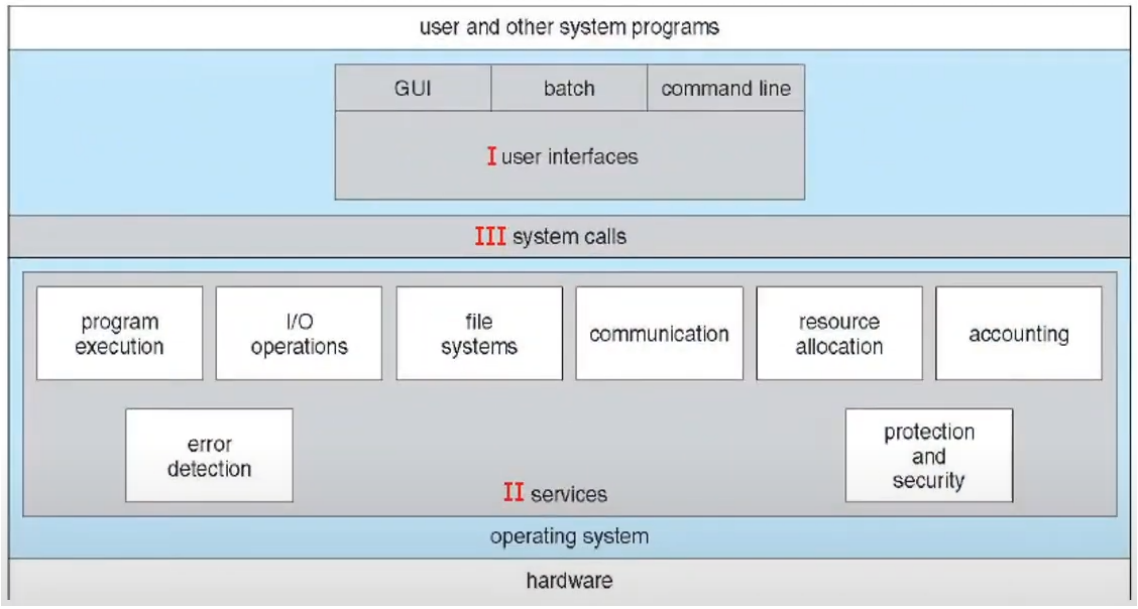

FIGURE 1 – SE

1. Command line (CLI)
2. Graphical user interface (GUI)
3. Interface Appel Système
4. Programme Système

1.1.1 Command line (CLI)

L'interface de ligne de commande (CLI) ou l'interpréteur de commandes permet une entrée de commande directe.
Ca consiste principalement à récupère une commande de l'utilisateur et l'exécuter.
Les commandes sont soit intégrées (implémentées dans le noyau), ou sont des noms de programmes systèmes.

1.1.2 Graphical User Interface (GUI)

C'est une interface conviviale dite "desktop".
Connection entre le souris, clavier et moniteur.
Les icônes représentent les fichiers, les programmes, les actions, etc...

Divers boutons de la souris sur les objets de l'interface provoquent diverses actions (fournir des informations, des options, exécuter la fonction, ouvrir le répertoire,...)

1.2 Les services

1.2.1 Les servics utilisateur

1. **Exécution d'un programme (Program execution) :** Le système doit être capable de charger un programme en mémoire, de l'exécuter, terminer son exécution (normalement ou anormalement) et indiquer s'il y avait des erreurs lors de l'exécution.
2. **Opérations d'entrées/sorties (I/0 operations) :** Un programme en cours d'exécution peut nécessiter des opérations d'E/S, ce qui peut impliquer un fichier ou un périphériques d'E/S
3. **Système de fichiers (File systems) :** Le système de fichiers présente un intérêt particulier. Les programmes doivent lire et écrire des fichiers et des répertoires, les créer et les supprimer, les rechercher, lister les informations du fichier, gérer les permissions.
4. **Communication :** Les processus peuvent échanger des informations, sur le même ordinateur ou entre ordinateurs sur un réseau. Les communications peuvent être effectuées via la mémoire partagée ou via le transfert de messages (paquets déplacés par le système d'exploitation)
5. **Détection d'erreurs (Error detection) :** Le système d'exploitation doit être capable de détecter, gérer et anticiper les différentes erreurs susceptibles de se produire dans le cpu, la mémoire, les périphériques d'E/S ou dans les programmes utilisateurs

1.2.2 Les services système

1. **Allocation de ressources (Ressource allocation) :** Lorsque plusieurs utilisateurs ou plusieurs jobs s'exécutent simultanément, des ressources doivent être allouées à chacun d'entre eux comme : les cycles cpu, RAM, périphériques d'E/S, etc.
2. **Comptabilité (Accounting) :** Pour garder une trace de qui (utilisateur) utilise quoi (ressource) et quand est-ce qu'il utilise et combien de fois, etc.
3. **Protection et sécurité (Protection and security) :** Protéger les informations stockées des accès non permis, gérer les processus concurrents, etc.

1.3 Les appels système

1.3.1 Les types d'appels système

1. **Contrôle de processus**

- Créer un processus, mettre fin à un processus
- Avorter processus
- Charger, exécuter
- Obtenir/définir des attributs de processus
- Attendre pour un certain temps
- Attendre/signaler un événement
- Gestion de la mémoire
- Debug
- Verrous pour gérer l'accès aux données partagées entre les processus
- etc.

2. **Gestion des fichiers**
   - Créer/supprimer les fichiers
   - Ouvrir/fermer les fichiers
3. **Gestion des appareils**
   - Demande, libération de l'appareil (request/release)
   - Lire, écrire, repositionner
   - Obtenir/définir les attributs de l'appareil
   - Connecter ou détacher des périphériques
4. **Communication**
   - Créer/supprimer les connexions
   - Transférer les informations d'état
   - Envoyer, recevoir des messages (communication par le modèle de passage de message)
   - Créer et accéder aux régions de la mémoire (communication par le modèle à mémoire partagée)
5. **Protection**
   - Contrôler l'accès aux ressources
   - Obtenir et définir des autorisations
   - Autoriser et refuser l'accès des utilisateurs à certaines ressources

1.3.2 Exemples d'appels sytème

	Windows	Unix
Process Control	CreateProcess()	fork()
	ExitProcess()	exit()
	WaitForSingleObject()	wait()
File Manipulation	CreateFile()	open()
	ReadFile()	read()
	WriteFile()	write()
	CloseHandle()	close()
Device Manipulation	SetConsoleMode()	ioctl()
	ReadConsole()	read()
	WriteConsole()	write()
Information Maintenance	GetCurrentProcessID()	getpid()
	SetTimer()	alarm()
	Sleep()	sleep()
Communication	CreatePipe()	pipe()
	CreateFileMapping()	shmget()
	MapViewOfFile()	mmap()
Protection	SetFileSecurity()	chmod()
	InitlializeSecurityDescriptor()	umask()
	SetSecurityDescriptorGroup()	chown()

FIGURE 2 – Exemples d'appels système

1.4 Interface Appel système

— Généralement, un numéro est associé à chaque appel système -> L'interface d'appel système maintient une table indexée selon ces chiffres

— L'interface d'appel système appelle l'appel système prévu dans le noyau du système d'exploitation et renvoie l'état de l'appel système et toutes les valeurs de retour

— L'appelant n'a pas besoin de savoir comment l'appel système est implémenté -> La plupart des détails de l'interface du système d'exploitation sont cachés du programmeur par l'API

— Interface de programmation pour les services fournis par le système d'exploitation
— Principalement accessible par les programmes via une interface de programme d'application (API) de haut niveau plutôt que par l'utilisateur directe des appels système
— Les 3 plus utiliser sont :
  — Win32 API pour Windows
  — POSIX API pour POSIX based systems (UNIX, Linux, and Mac OS X)
  — Java API for the Java virtual machine (JVM)

As an example of a standard API, consider the `read()` function that is available in UNIX and Linux systems. The API for this function is obtained from the `man` page by invoking the command

```
man read
```

on the command line. A description of this API appears below:

```
#include <unistd.h>

ssize_t    read(int fd, void *buf, size_t count)
```

return value — function name — parameters

A program that uses the `read()` function must include the `unistd.h` header file, as this file defines the `ssize_t` and `size_t` data types (among other things). The parameters passed to `read()` are as follows:

- `int fd`—the file descriptor to be read
- `void *buf`—a buffer where the data will be read into
- `size_t count`—the maximum number of bytes to be read into the buffer

On a successful read, the number of bytes read is returned. A return value of 0 indicates end of file. If an error occurs, `read()` returns −1.

FIGURE 3 – Example of standard API

Interruption :

— Une interruption cause un transfert de contrôle à la routine de service d'interruptions et indique au système de gérer et réagir à des événements d'E/S

— Après la gestion de l'interruption, le cpu revient à l'instruction interrompue en chargeant son adresse qu'il avait enregistrée avant de céder la main à la routine de service d'interruptions

— Un **"trap"** ou une **exception** est une sorte d'interruption générée par un software causée par une erreur ou une requête d'appel système

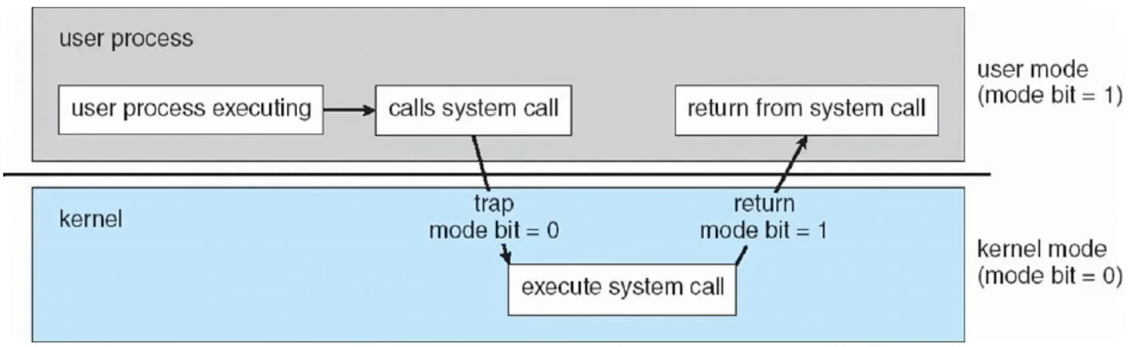

FIGURE 4

Pour éviter les boucles infinies, le noyau doit faire attention à :

— Utiliser un timer pour générer une interruption dans le futur

— Avant de retourner en mode user, le SE arme le timer (instructions privilégiées non accessibles en mode utilisateur)

— Quand le timer sonne, l'interruption redonne la main au noyou

1.5 Programmes Système

Les programmes système (utilitaires) fournissent un environnement pratique pour le développement et l'exécution du programme.
Certains sont simplement des interfaces pour les appels système, d'autres sont considérablement plus complexes :

— Manipulation de fichier

— Informations d'état

— Support de langage de programmation

— Communications

— Service d'arrière plan

La vue de la plupart des utilisateurs du système d'exploitation est définie par les programmes système et les programmes d'application, et non par les appels systèmes

réels.

Manipulation de fichier :
Créer, supprimer, copier, renommer, imprimer, vider, répertorier et manipuler généralement des fichiers et des répertoires (touch, mkdir, cp, ...), éditeur de texte (vim, emacs, nano), Commandes spéciales pour rechercher le contenu des fichiers ou effectuer des transformations du texte (grep, sed)

Informations d'état :
date, heure, quantité de mémoire disponible, espace disque, nombre d'utilisateurs, etc.

Support de langage de programmation :
Compilateurs, assembleurs, débogeurs parfois fournis (gcc, gdb, python)

Service d'arrière plan :
Lancement au démarrage : soit uniquement pour le démarrage du système puis ils se termine ou du démarrage du système à l'arrêt, Fournir des fonctionnalités telles que la vérification du disque, l'impression, etc...