Task 3: Adding System Calls to XV6

Jana Saleh 900204192
Muhammad Azzazy 900202821
Mariam Dahab 900192441

Table of contents

01

System calls description

02

Reasons why they are system calls

03

How the system calls were added to xv6

04

User programs that use the system calls

01 System Calls Description

numofprocs()	- iterates over the process table and checks on the process state and if it’s not unused it counts it. -returns the number of process in the system
getchild()	-iterates over the process table and access the parents to check over the pid and compares it to the id of my process and counts it -returns the number of child process in the system
cps()	-iterates over the process table and prints the name , id , and current state of active process - returns any integer other than 1 to indicate successfulness
date()	Uses cmostime to get the struct of type rtcdate (real-time clock date) which contains the current year, month, day, hour, minute and second.

02

Reasons Why They are System Calls

Reasons they are system calls

1. **numofprocs() & getchild()** → they are system calls because they are useful and allow the user to know the state or the number of process and child process in their system and havings is insights regarding the child processes running or all process in general.

1. **cps()** → it prints to the user all the active process which requires access to the process table and this is a feature available only for the kernel.

1. **date()**→takes as input a reference to a struct of type rtcdate and modifies it. This is done using the complementary metal-oxide semiconductor time which is part of the basic input-output system. Therefore, this function should only be in kernel space.

03

How System Calls were Added

Changes were made in :

- MakeFile
- defs.h
- proc.c
- syscall.c
- syscall.h
- user.h
- usys.S
- sysproc.c

proc.c	Implemented specific system calls function here in order to access the ptable because it cannot be accessed outside this file (for process related system calls)
syscall.c	➔ Added system calls to the array of function pointers for system call functions ➔ Added all system calls functions prototype
syscall.h	Define the system calls using numbers to index the array of function pointers
usys.S	Added the interface of the system call for the user program to be able to call the system call in the program
user.h	Adding the system calls functions prototype which will be called by the user program
sysproc.c	Implemented the system call in this file
Makefile	Added the user programs to test the system calls

04

User Programs That Use The System Calls

NumOfprocs() and getchild() User Program

The user program gives the user 4 choices whether to

1. return root process which is always
2. Return number of child processes
3. Return number of all processes
4. Exit the program

PS User Program

- The ps program calls cps() to print the process information into the terminal when user types ps .

Date User Program

This user program takes no command-line arguments.

1. It calls the date system call and passes a struct of type rtcdate to check whether it returns successfully. If it does not it prints an error message and exits.
2. If there were no errors, it proceeds to manipulate the member variables of the struct by doing the following:
   a. Uses a variable called month to store the string literal that corresponds to the number stored in month through conditional statements.
   b. The index of the day of the week is calculated using a certain formula.
   c. Uses a variable called day to store the string literal that corresponds to the day of the week through conditional statements.
3. Finally, it prints the real-time date & time in the following format:
   <day_of_week> <month> <hours>:<minutes> EET <year>

Thanks!

Do you have any questions?
janasaleh@aucegypt.edu
mhdahab@aucegypt.edu
muhammad-azzazy@aucegypt.edu

CREDITS: This presentation template was created by **Slidesgo**, including icons by **Flaticon**, infographics & images by **Freepik**