**Proiect PROIECTAREA SISTEMELOR DE OPERARE**

Rezultate parțiale

Îndrumător de proiect: Slt. Vaman Adina

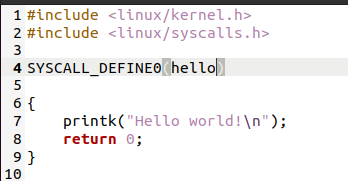
Sd. sg. Corbu Ionuț-Daniel

Sd. sg. Toader Radu Marian

**Task-uri rezolvate și task-uri propuse**

În elaborarea proiectului “Apeluri de sistem pentru aflarea de informații din sistemul de operare” am rezolvat următoarele task-uri intermediare:

* ne-am familiarizat cu fișierele care sunt compilate pentru obținerea imaginii kernelului (vmlinuz file);
* am exersat modalitatea în care sunt concepute apelurile de sistem, implementând un apel de sistem care afișează în kernel log;am creat un director pentru apelul de sistem și o sursă .c:



* am creat un fișier Makefile pentru compilarea sursei:



* am modificat Makefile-ul din directorul kernelului pentru a menționa directorul corespunzător apelului:



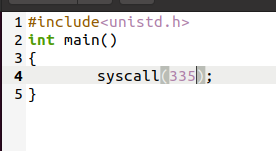
* am modificat headerul ./include/linux/syscalls.h care conține definiția apelurilor de sistem astfel încât să conțină și definiția apelului nostru:



* am modificat tabela ./arch/x86/entry/syscalls/syscall\_64.tbl pentru a adăuga o intrare pentru apelul nostru de sistem:



* am compilat kernelul de Linux cu apelurile de sistem adaugate și am testat funcționalitatea apelului de sistem folosind o sursă .c și utilitarul dmesg care întoarce mesajele din kernel space:



După compilarea sursei și executarea executabilului, putem folosi dmesg pentru a observa mesajul trimis din kernel space:



* ne-am documentat implementarea apelurilor de sistem care să primească parametrii în așa fel încât să putem afla informații despre o anumită instanță pe care ne-o propunem spre analizare (un proces) ;

*SYSCALL\_DEFINEn(nume\_apel,tipul argumentului 1,numele argumentului 1,etc)*

* ne-am documentat în privința structurii task\_struct;
* am luat în calcul inspectarea stack-ului pentru fiecare proces astfel încât să distingem zonele alocate fiecărui proces;
* am luat în calcul posibilitatea de a trasmite ca parametru un PID și de a ni se întoarce starea procesului corespunzător;
* am luat în calcul posibilitatea de a transmite ca parametru un PID pentru a afla părintele său;
* am luat în calcul să inspectăm files\_struct pentru a afla ce fișiere deschide procesul al cărui PID îl primim ca parametru etc.

***Probleme întâmpinate***

Compilarea kernelului a fost foarte anevoioasă din cauza numărului de core-uri implicate în acest proces și din cauza duratei foarte mari. De asemeneam, erorile întâlnite după mai multe compilări au încetinit procesul de lucru, însă odată cu bootarea pe versiunea de kernel corespunzătoare, compilarea nu mai presupune apariția unor erori neașteptate