Prvi domaći zadatak iz predmeta Napredni operativni sistemi

Za zadatak sam imao prvu temu koja zahteva da se za prosleđeni proces identifikator odštampa DFS (preorder) stablo sa korenom u tom procesu, sa osnovnim informacijama iz task\_struct strukture. Zadatak je implementiran tako da se štampaju identifikator procesa i polje comm iz task\_struct strukture. Zahtev zadatka je bio da se navedena funkcionalnost implementira i preko sistemskom poziva i preko kernel modula. Prvo će biti razmotreno rešenje preko kernel modula.

Rešenje preko kernel modula

Rešenje preko kernel modula se sastoji iz tri fajla, **kmodule.c**, **test.c** i **Makefile**. Fajl kmodule.c sadrži kod vezan za kernel modul. Na početku ovog fajla prvo su navedene sve neophodne header datoteke. U nastavku fajla date su tri metode za rad sa stekom, stek je ovde nephodan zbog obilaska stabla, kako se ne bi radilo sa rekurzivnim pozivima. U nastavku dat je poziv makroa module\_param zato što je neophodno da ovaj modul ima jedan parametar a to je identifikator procesa za koji se štampa DFS obilazak. Zatim su date kmodule\_enter i kmodule\_exit procedure koje se redom izvršavaju kod učitavanja i kod odbacivanja kernel modula. Sav kod koji radi DFS obilazak dat je u kmodule\_enter proceduri. Ideja je da se prvo prođe kroz listu procesa kako bi se našao proces sa prosleđenim identifikatorom, a kada se proces nađe, pomoću steka se uradi DFS obilazak stabla procesa u čijem je korenu dati proces. Drugi fajl (test.c) se samo koristi za testiranje, gde se na osnovu njega (kada se pokrene test.o) napravi stablo do 8 procesa koje se kasnije obilazi. Treći fajl (Makefile) sadrži dva recepta, od kojih je prvi preuzet sa interneta. Drugi recept učitava kernel modul, odbacuje ga i na kraju štampa dnevnik poruka komandom dmesg. Bitno je napomenuti da se kod testiranja make komanda aktivira sa:

|  |
| --- |
| make test PID=<identifikator\_procesa> |

Za proveru korektnosti rezultata može se koristiti komanda (koja vizuelizuje stablo procesa):

|  |
| --- |
| pstree -p |

Celokupni workflow bi izgledao ovako:

|  |
| --- |
| gcc -o test.o test.c |
| ./test.o & |
| make |
| pstree -p*;ovde pogledamo pid koji nas interesuje* |
| make test PID=20039 |
| *;proverimo korektnost rezultata* |

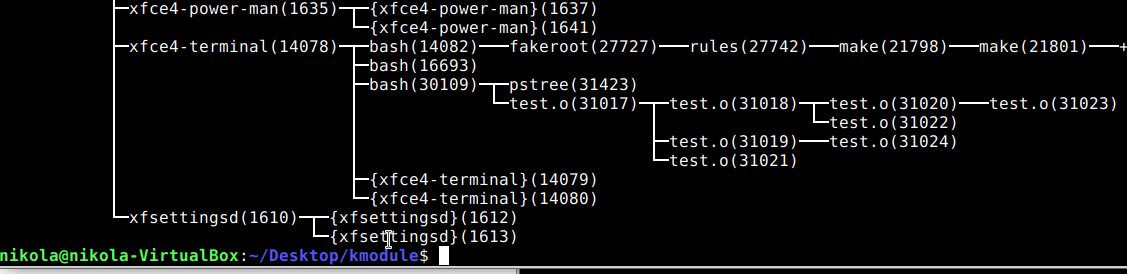
Rešenje preko sistemskog poziva

Za rešenje preko sistemskog poziva iskorišćen je kod iz kernel modula ali je izmenjen na takav način da omogućava da se data procedura pozove kao sistemski poziv. Za to je prilikom deklaracije sistemskog poziva iskorišćen makro SYSCALL\_DEFINE1. Za implementaciju ovog rešenja korišćen je tutorijal za dodavanje novog sistemskog poziva. Za testiranje koristi se fajl test\_syscall.c, gde će test\_syscall.o kada se pozove sa jednim argumentom komandne linije koja predstavlja identifikator procesa, obaviti sistemski poziv sa tim argumentom. Za testiranje se isto kao u prethodnom slučaju koristi fajl test.c koji ima zadatak da napravi stablo od osam procesa.

|  |
| --- |
| ./test.o & |
| pstree -p |
| ./test\_syscall.o 20039 |

Primeri rada

Na sledećim slikama dati su primeri rada. U korenu stabla koje se obilazi nalazi se proces sa identifikatorom 31017.



Prva kolona predstavlja identifikatore procesa, a druga predstavlja polje comm iz task\_struct.

