Warszawa, 05.11.2019

student: Patryk Jan Sozański

grupa: 2I5

nr albumu: 300258

SYSTEMY OPERACYJNE: LABORATORIUM NR 1

RAPORT Z WYKONANEGO ĆWICZENIA

**Treść zadania**

Utworzyć usługę systemową, która przyjmie jeden argument (np.: liczba 32 bitowa) i ten argument w swojej implementacji dodaje do numeru procesu ją wywołującego.

**Dodawanie nowego wywołania systemowego**

W celu dodania nowego wywołania systemowego o nazwie do\_addtoid wykonałem następujące czynności:

* w pliku /usr/include/minix/callnr.h zwiększyłem stałą N\_CALLS o jeden i dodałem na końcu identyfikator nowego wywołania systemowego ADDTOID:

#define ADDTOID 77

* napisałem procedurę obsługi do\_addtoid i umieściłem ją w pliku /usr/src/mm/main.c:

int do\_addtoid (void){

if(((mproc[who].mp\_flags & IN\_USE) != 0) && (who >= 0))

return (mproc[who].mp\_pid + pid);

else

return ENOENT:

}

gdzie:

mrpoc – struktura zawierająca informacje zarządzania pamięcią dla każdego procesu (opis w pliku /usr/src/mm/mproc.h)

who – numer procesu procesu wywołującego (opis w pliku /usr/src/mm/main.c)

pid – synonim zmiennej mm\_in.m1\_i1 (opis w pliku /usr/src/mm/param.h)

ENOENT – błąd (opis w pliku /usr/include/errno.h)

* w pliku /usr/src/mm/proto.h umieściłem prototyp powyższej funkcji:

\_PROTOTYPE( int do\_addtoid, (void) );

* w pliku /usr/src/mm/table.c w tablicy call\_vector w odpowiednim miejscu (po adresie nr 76) wstawiłem adres funkcji do\_addtoid:

do\_addtoid, /\* 77 = ADDTOID \*/

* w pliku /usr/src/fs/table.c w tablicy call\_vector w odpowiednim miejscu (po adresie 76) wstawiłem adres pusty funkcji:

no\_sys, /\* 77 = ADDTOID \*/

* wykonałem rekompilację i przeładowanie systemu Minix z nowym jądrem. W tym celu wywołałem następujące polecenia:

# cd /minix

# rm\*

# cd /usr/src/tools

# make clean

# make hdboot

# cd

# shutdown

# boot

* w pliku /root/main.c umieściłem program testujący poprawność działania zaimplementowanego wywołania systemowego:

#include <stdio.h>

#include <stdlib.h>

#iinclude <unistd.h>

#include “/usr/include/lib.h”

#include “/usr/include/minix/type.h”

int main (int argc, char\* argv[]){

int result;

int pidd;

message msg;

int value = atoi(argv[1]);

msg.m1\_i1 = value;

if(argc == 1)

printf(“Argument needed!\n”);

else{

pidd = getpid();

result = \_syscall(0, ADDTOID, &msg);

printf(“Given value = %d\nPID = %d\n”, value, pidd);

printf(“Process index + given value = %d\n”, result);

}

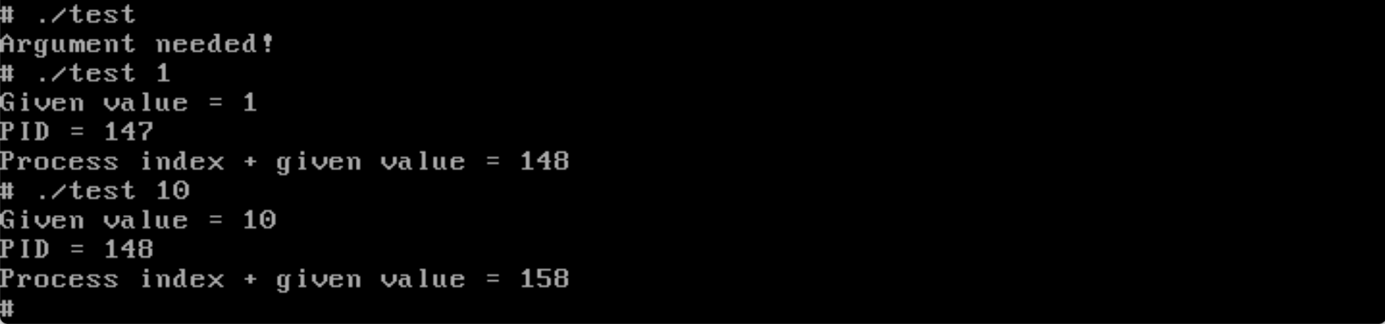
return 0;

}

gdzie:

getpid() – funkcja z biblioteki unistd.h zwracająca ID procesu (PID) procesu wywołującego

* po wywołaniu powyższej funkcji bez argumentu, z argumentem „1” i z argumentem „10” otrzymałem następujący wynik:



Widać, że wywołanie systemowe poprawnie przyjmuje jeden argument typu int i zwraca liczbę równą sumie tego argumentu i numeru procesu go wywołującego, co było celem ćwiczenia.