Kompilacja jądra i wywołania systemowe

Jakub Robaczewski

`/usr/src/mm/proto.h`
Dodałem prototyp funkcji.

`/usr/src/mm/table.c`
`/usr/src/fs/table.c`
Dodałem odwołania do funkcji i adres pusty.

`/usr/include/minix/callnr.h`
Dodałem definicję nowego odwołania systemowego.

`/usr/src/tools/main.c`
Dodałem procedurę obsługi do nowego wywołania systemowego. Funkcja iteruje po tablicy procesów (ich liczba zapisana jest pod makrem NR_PROCS) sprawdzając czy proces jest używany (flaga IN_USE) oraz, czy jego PID jest taki sam jak podany. Jeśli oba warunki są spełnione zwraca numer procesu, w przeciwnym wypadku zwraca kod błędu ENOENT. Argument wejściowy przekazywany jest za pomocą struktury mm_in.

`/root/test.c`
Funkcja testująca realizowana jest za pomocą podfunkcji getprocnr(), która pobiera numer PID procesu i wywołuje wywołanie, które zdefiniowałem w poprzednim kroku, podając jego kategorię (MM), numer (78) i wskazanie na strukturę wejściową do której wpisuje argument. Główna funkcja iteruje po pętli rozpoczynającej się od wskazanego identyfikatora i przechodzącej do następnych 10 identyfikatorów. Dla każdego identyfikatora funkcja wywołuje podfunkcję i w zależności od jej wyniku wyświetla numer albo kod błędu.
Funkcję można wywołać poleceniem:
`/root/test [identyfikator początkowy]`