/usr/include/minix/callnr.h
Dodanie identyfikatora nowego wywołania.

```
#define GETPROCNR      69
```

/usr/src/mm/main.c
Dodanie funkcji zwracającej indeks procesu, albo błąd ENOENT w przypadku jego braku

```
PUBLIC int do_getprocnr(void) {
        int procid = mm_in.m1_i1;
        int idx = 0;
        for(idx ; idx< NR_PROCS ;++idx){
                if((mproc[idx].mp_pid == procid) && (mproc[idx].mp_flags & IN_USE))
                        return idx;
        }
        return ENOENT;
}
```

/usr/src/mm/proto.h
Dodanie prototypu funkcji do_getprocnr

```
_PROTOTYPE( int do_getprocnr, (void) );
```

/usr/src/mm/table.c
Dodanie adresu funkcji

```
    do_getprocnr,    /* 69 = getprocnr    */
```

w src/fs/table.c nie zmieniono nic, ponieważ już znajdowało się tam no_sys

Program testujący pobiera nr procesu który ma zweryfikować i zwraca nr procesu w tabeli, albo ERRNO gdy proces nie wystepuje.

```c
#include "src/lib/other/syscall.c"
#include "include/minix/type.h"
#include <stdio.h>
#include <stdlib.h>


int getprocnr(int ident_pid)
{
  message msg;
  msg.m1_i1 = ident_pid;
  return _syscall(MM,GETPROCNR,&msg);
}

int main(int argc, char *argv[])
{
  if(argc < 2)
  {
    printf("Not enough arguments\n");
    return 1;
  }
  else
  {
    int i,j,idx;
    for(i = 1; i < argc; ++i)
      for(j = 0; j < 10; ++j)
        if((idx = getprocnr(atoi(argv[i])+j)) > 0)
        printf("PID : %d nr procesu w tabeli mproc %d\n",(atoi(argv[i])+j),idx);
        else
        printf("PID : %d ERRNO: %d\n",(atoi(argv[i])+j), errno);
   }
   return 0;
}
```