Payment System in MINIX

The goal of the task is to enable processes in the MINIX system to have money and perform mutual transfers.

Each process receives INIT_BALANCE units of currency at the start. Then processes can transfer money to each other, i.e., process P can initiate a transfer of n currency units to process Q. For the transfer to succeed, process P must have at least n currency units on its account (the process's account balance cannot be negative), and process Q must have no more than MAX_BALANCE - n currency units on its account. Additionally, as a basic safeguard against money laundering, we require that processes P and Q are not in a parent-child relationship. If the transfer is successful, the account balance of process P decreases by n currency units, and the account balance of process Q increases by n currency units.

The money of processes is not inherited - when a process terminates, the currency units accumulated by it disappear.

Note: granting each process new currency units at the start inevitably leads to inflation, but let's leave this problem to economists. New System Call

The task involves adding a new system call PM_TRANSFER_MONEY and a library function int transfermoney(pid_t recipient, int amount). The function should be declared in the file unistd.h. Constants INIT_BALANCE = 100 and MAX_BALANCE = 1000 should be defined in the minix/config.h file.

The function int transfermoney(pid_t recipient, int amount) should perform a transfer of amount currency units from the account of the process calling the function to the account of the process with the identifier recipient. Upon successful completion of the transfer, the function returns the account balance of the calling process after the transfer.

Note: a process can check its account balance by, for example, transferring 0 currency units to itself.

If the transfer fails, the function transfermoney returns -1 and sets errno to the appropriate error code:

- if recipient is not the identifier of the currently executing process, errno is set to ESRCH;
- if recipient is the identifier of a process that is a descendant or ancestor of the process calling the transfermoney function, errno is set to EPERM;
- if the value of amount is negative, or the process calling transfermoney has fewer than amount currency units on its account, or the process with the identifier recipient has more than MAX_BALANCE - amount currency units on its account, errno is set to EINVAL.

The operation of the transfermoney() function should involve the use of the new system call PM_TRANSFER_MONEY, which needs to be added to the PM server. A custom message type should be defined to pass parameters.

Solution Format

Below, we assume that ab123456 represents the identifier of the student solving the task. You should prepare a patch with changes in the /usr directory. Obtain a file containing the patch named ab123456.patch using the command:

```
diff -rupNEZbB original-sources/usr/ my-solution/usr/ > ab123456.patch
```

where original-sources is the path to the unchanged MINIX sources, and my-solution is the path to the MINIX sources containing the solution. This diff command will recursively scan files in the original-sources/usr path, compare them with files in the my-solution/usr path, and generate a file named ab123456.patch summarizing the differences. This file will be used to automatically apply changes to a clean copy of MINIX, where tests will be conducted. More information about the diff command can be found in the manual (man diff).

Placing the patch in the / directory on a clean copy of MINIX and executing the command:

```
patch -p1 < ab123456.patch
```

should result in applying all the expected changes required by the solution. Make sure the patch contains only necessary differences.

After applying the patch, the following commands will be executed:

```
make && make install in the /usr/src/minix/fs/procfs, /usr/src/minix/servers/pm,
/usr/src/minix/drivers/storage/ramdisk, /usr/src/minix/drivers/storage/memory,
and /usr/src/lib/libc directories,
make do-hdboot in the /usr/src/releasetools directory,
reboot.
```

The solution in the form of a patch ab123456.patch should be uploaded to Moodle. Remarks

- The PM server stores information about processes in the mproc array declared in the mproc.h file.
- It is worth analyzing how PM implements system calls. More information about the operation of this server will be provided in laboratory 8.
- The task mentions executing the make command in the /usr/src/minix/fs/procfs, /usr/src/minix/drivers/storage/ramdisk, and /usr/src/minix/drivers/storage/memory directories because they contain files including mproc.h.
- You should independently test the solution. One of the basic scenarios is as follows: run process A, which then launches processes B and C; processes B and C start transferring money to each other.
- Points will not be awarded for a solution where the patch does not apply correctly, does not compile, or causes a kernel panic when the system is booted.