

# Vežbe 2

- Teme koje ćemo preći:
  - Osnovno o sistemskim pozivima
  - obrada gresaka (errno, strerror, perror, \_\_FILE\_\_, \_\_func\_\_, \_\_LINE\_\_)
  - Rad sa fajlovima:
    - open() / creat()
    - read()
    - write()
    - lseek()
    - unlink()
    - mkdir()

# Sistemiški pozivi - osnovno

- Operativni sistem je “menadžer” resursa računara! [GUI ! = OS]
- Kada pristupamo resursima računara moramo koristiti sistemske pozive [npr. kada pišemo u fajl podaci se smeštaju na disk i to reguliše OS]
- Funkcije standardne C biblioteke koriste sistemske pozive u svojoj implementaciji [npr. na UNIX zasnovanim sistemima *printf()* koristi *write()*, *fscanf()* koristi *read()*, *fopen()* koristi *open()* itd.]
- **Sistemiški pozivi se izvršavaju van vašeg programa - izvršava se direktno mašinski kod OS-a.**
- **Sama procedura zvanja sistemskih poziva se implementira u assembleru.**

# Obrada grešaka

- Jedan od najbitnijih aspekata dizajna programskog jezika i softverskih rešenja
- Dva najčešća modela:
  - Izuzeci (eng. exceptions) [C++, Java, C#, Python, PHP, Javascript itd.]
  - Kodovi grešaka (eng. error codes) [C, C++]
- S obzirom na to da se u C-u koriste kodovi grešaka tome ćemo posvetiti više pažnje

# Kodovi grešaka

- U C-u sve funkcije standardne C biblioteke postavljaju kodove greške
- Kodovi se nalaze u promenljivoj *errno* [`#include <errno.h>`]
- Ukoliko C f-ja uspe *errno* se postavlja na SUCCESS, inače na kod greške
- String za kod greške -> *strerror()*
- Ispis greške -> *perror()* ili *fprintf(stderr, "%s\n", strerror(errno))*

# Prijava grešaka

- Greške se moraju obrađivati! [exception-i su smišljeni da spreče programere da zanemare greške]
- Prilikom ispisa greške zgodno je koristiti predefinisane makroe koji postoje u C-u:
  - `__FILE__` - ime fajla [const char \*]
  - `__func__` - ime funkcije [const char \*]
  - `__LINE__` - broj linije u fajlu [int]
- Prilikom obrade grešaka postoje 2 mogućnosti:
  - ispisati poruku o grešci [ovaj pristupćemo koristiti na kursu jer je jednostavniji, a naši programi su mali]
  - obraditi grešku [čak je i SEGFAULT moguće obraditi i često uspešno oporaviti program]

# Sistemi pozivi - praktično

- Detaljne informacije o svakom sistemskom pozivu mogu da se vide u sekciji 2 man strana:
  - man 2 read
  - man 2 write
  - man 2 open itd.
- Navedeni sistemski pozivi u C-u su zapravo GLIBC omotači pravih sistemskih poziva koji su implementirani u assembleru, a umetnuti u C kod. [prikazaćemo to za X86 arhitekturu]
- Napomena za pozive *read()* i *write()*:
  - rade sa bajtovima ne sa C stringovima!
  - dakle, kad nešto pročitamo sa *read()* napunjeni bafer nije terminiran nulom!
- **Primeri upotrebe navedenih sistemskih poziva...**