Povezivanje

- U principu, prema tome, linker može da prijavi samo dve vrste grešaka:

1. *Simbol nije definisan*: kada napravi evidenciju (u svojoj tabeli simbola) o tome koji fajlovi izvoze (definišu) koje simbole, a koji fajlovi uvoze koje simbole, linker može da zaključi da je neki fajl tražio (uvezao) neki simbol koji nijedan fajl nije definisao (izvezao); u informaciji o ovoj grešci linker ne može da kaže ništa o tome šta je simbol bio u izvornom programu niti gde je u izvornom kodu tražen (jer te informacije po pravilu i nema); tipični uzroci jesu:
   - zaboravljena definicija neke deklarisane funkcije ili objekta (retko, čista omaška)
   - zaboravljena neki *obj* ili *lib* fajl na spisku za linkovanje (omaška)
   - neka povezana biblioteka zavisi od (uvozi simbole iz) neke druge biblioteke, koja nije povezana
2. *Simbol višestruko definisan*: kada naiđe na simbol koji neki fajl izvozi, a isti taj simbol već postoji u tabeli simbola jer ga je neki drugi fajl već izvezao, linker prijavljuje grešku; ovo je situacija tzv. *sukoba imena* (*name clash*): dva izvorna fajla definisala su ista globalna imena sa eksternim vezivanjem (primetiti to da imena sa internim vezivanjem nisu ni vidljiva linkeru i ne mogu da naprave sukob); tipični uzroci jesu:
   - sukob imena u korisničkom programu
   - sukob imena iz korisničkog programa sa imenom koje je izvezla biblioteka

   Ovakve pojave, tj. sukobe imena, značajno smanjuje korišćenje koncepta *prostora imena* (*namespace*) na jezik C++, čime se izbegava potreba za globalnim imenima i obeshrabruje njihova upotreba

Pretprocesiranje

- Posledica: za svaki entitet definisan u fajlu *A* koji treba da se koristi u drugim fajlovima *B* moraju postojati deklaracije u svim tim fajlovima *B*
- Ali šta se dešava ako te deklaracije nisu identične, npr. kao posledica greške? Na primer, promenjena je projektna odluka da promenljiva *x* ne bude tipa *int* nego tipa *double*, ali izmena nije urađena svuda:

```
// A.cpp

double x = 2.3;
```

```
// B.cpp

extern int x;
void g () {
   ...x++...
}
```

- Prevodilac neće prijaviti grešku, jer je za njega kod u fajlu *B.cpp* ispravan; još gore, on će prevesti operacije sa *x* kao operacije sa celobrojnom promenljivom, a ne kao sa racionalnim brojem - potpuno drugačija implementacija i tretman binarnog sadržaja (može biti i različite veličine)
- Grešku neće prijaviti ni linker, jer je simbol *x* korektno izvezen i uvezen, osim ako prevodilac u generisani simbol (kao niz znakova) ne enkoduje i informaciju o tipu (što se često radi), npr. ako za ovaj *x* u zaglavlju *obj* fajla napravi simbol poput *x@int*
- Problem je nastao zbog prekršaja principa lokalizacije projektne odluke: manifestacije odluke da je *x* određenog tipa raštrkane su po kodu i redundantno kopirane kao deklaracije tog *x*; u slučaju potrebe za promenom, može se dogoditi ovakva greška zbog nekonzistentne izmene