10.3 디버거

- gdb
  - GNU debugger – 심볼릭 디버깅 지원
- 디버거용 컴파일 옵션 : -g
  - 직접 실행파일 생성

```
$ cc -g opt.c -o opt
$ cc -g opt2.c reverse.c -o opt2
```

  - 다중 모듈 프로그램 컴파일
    - 오브젝트 파일 생성할 때에 –c와 함께 –g 옵션 사용
      $ cc –g –c reverse.c
      $ cc –g –c opt2.c
    - 오브젝트 파일을 링크할 때에는 –g 옵션 불필요
      $ cc reverse.o opt2.o –o opt2

디버깅을 위한 makefile

```
CFLAGS = -g
OBJS = opt2.o reverse.o

opt2 : $(OBJS)
        cc $(OBJS) -o opt2
opt2.o : opt2.c reverse.h
reverse.o : reverse.c reverse.h
```


gdb 주요 명령어

- gdb 명령어 실행
  $ gdb *exefile*
  (gdb) ...
- gdb 주요 명령어 – 교과서 표 참조
- 실행/종료
  - run 인수 … 실행
  - quit … 종료
- 소스코드 보기
  - list (l) … 현재위치부터 10줄 출력
  - list 행번호/함수
  - list 파일이름:행번호/함수
- 정지점(breakpoint)
  - break (b) 행번호/함수 … 설정
  - clear (cl) 행번호/함수 … 제거
  - delete (d) … 모든 정지점 제거

- 데이터 출력
  - print (p) 수식 … 정지점에서 출력
  - print /x 수식 … 16진수 출력
    - 형식 : x, d, u, o, t (2진수), a(주소)
- 연속 실행
  - continue (c) … 다음 정지점까지 실행
- 자동 데이터 출력
  - display (disp) 수식
  - display /x 수식
- 정지점 관련 추가 기능
  - info break … 현재 설정된 정지점
  - condition (cond) 정지점번호 조건식 … 정지점의 정지 조건
  - disable (dis) b .. … 정지점 비활성화
  - enable (en) b .. … 정지점 활성화

- 데이터 정지점
  - watch (wa) 변수/수식 .. 값이 변경되면 정지
- 프로그램 실행 제어
  - next (n) … 한 줄 단위 실행 (함수 호출도 한 줄로 처리)
  - step (s) … 함수 호출은 호출되는 함수로 들어감
  - finish (fin) … 현재 함수 끝까지 실행
  - until (u) … 임시 정지점(한 번만 사용)
- 함수 호출 역추적
  - backtrace (bt) … 프로그램의 스택 프레임 정보
  - up … 현재 함수를 호출한 함수
  - down … 현재 함수가 호출한 함수
- 변수 값 변경 : 프로그램 실행에 영향을 줌
  - print a=100
  - set var a=200

10.4 프로그래밍 관련 유틸리티

- gprof - 프로그램 수행 감시
  - 함수의 호출 횟수와 소요 시간을 수집하여 출력
  - 성능 개선에 도움을 줌
- gprof 준비 및 실행 과정
  1. –pg 옵션을 사용하여 컴파일 (object 파일 생성 및 링크 시에 모두 사용)
  2. 프로그램 실행 : profile용 파일인 gmon.out 파일 생성
  3. gprof 실행 :

     $ gprof execfile [gmon.out] … verbose output

     $ gprof -b execfile [gmon.out] … brief output
- splint – C프로그램 검사
  - 프로그램 실행 시 발생할 수 있는 잠재적 오류 검사

10.5 라이브러리

- 라이브러리
  - 정적 라이브러리 – 링크할 때 실행파일에 포함됨
  - 동적 라이브러리 – 공유 라이브러리 사용, 실행할 때에 동적으로 링크
    최근에 주로 사용
- ar – 정적 라이브러리 생성

```
$ ar rv libmy.a mystrcpy.o
```

- 정적 라이브러리 사용

```
$ cc -o my my.c libmy.a          로컬 라이브러리
$ cc -o my my.c -lname           표준 라이브러리 디렉토리
$ cc my.c -L. -lmy -o my         라이브러리 디렉토리 추가
```


- nm – 실행파일에 포함된 심볼 출력
- strip – 실행파일에 포함된 심볼 제거
- readelf – 실행파일(elf 파일) 정보 출력
- size – 실행파일 섹션 크기

10.6 소스코드 관리 시스템

- 소스코드 관리
  - 개발과정에서 작성한 모든 버전을 저장하고, 접근 및 보호 수행
- 소스코드 관리 시스템 종류
  - SCCS (source code control system)
  - RCS (revision control system)
  - CVS (concurrent versions system) … 중앙서버 사용
  - SVN (subversion) … 중앙서버 사용
  - Git … 분산 서버 사용

RCS 사용하기

- RCS 디렉토리 생성

  $ mkdir RCS

- check in

  $ ci file // RCS 디렉토리에 file 대한 RCS file 생성, 현재 file 삭제함

  $ ci –u file // 현재의 file을 그대로 둠 (unlock) - 읽기전용

  $ ci –l file // 현재의 file을 그대로 둠 (lock 상태) - 계속 편집가능

  $ ci –r2.0 file // file의 버전을 2.0으로 하여 저장함

  - 버전은 1.1부터 생성됨, check in 할 때마다 버전이 갱신됨

- check out

  $ co file // RCS 디렉토리에서 file의 최신 버전을 가져와서
  // working file로 저장함 (read only)

  $ co –l file // locking 상태로 check-out. 다른 사람들이 사용하지 못함

  $ co –r1.1 file // file의 1.1버전을 가져옴

- log 보기

  $ rlog file // file이 revision 정보를 보여줌