소프트웨어공학과

리눅스 응용

기말고사 대비

시스템 프로그래밍 리눅스&유닉스. 7 장~12 장

이시헌

2023-6-14

교재의 객관식 연습문제 정리

- 7장: 프로세스 생성과 실행

1. fork( ) 함수는 포크를 수행하고 부모와 자식에게 각각 다른 리턴을 수행한다.

```
pid_t pid;
pid = fork( );

if(pid == 0) //자식프로세스에서 0리턴, 부모프로세스에서는 자식프로세스의 pid 리턴.
```

2. 프로그램 종료시의 행동 지정.
   A. exit( ) : 프로그램 종료. 인자는 종료시 넘길 상태값.
   B. 프로그램 종료시 작업 예약 함수
      i. atexit( ) : 인자로 지정하는 함수의 인자와 리턴값이 없을 때 사용.
      ii. on_exit( ) : 인자로 지정하는 함수에 넘겨줄 인자 지정 가능.

```
atexit(cleanup1( ));                    //인자와 리턴값 없는 함수 실행 예약.
on_exit(cleanup2, (void *)20);
//인자가 있는 함수 예약. 예약함수의 인자를 두번째 인자로 넘길 수 있다.

exit(0);   //프로그램 종료.
```

3. 현재 프로그램을 다른 명령으로 바꿔서 실행.
   A. exec함수군으로 다른 프로그램을 exec함수를 호출함 프로그램의 메모리 공간에 덮어쓸 수 있다. (함수 실행 즉시 원래 프로그램은 없어진다. 이는 fork( )로 생성된 자식 프로세스에 새로운 프로그램을 실행할 때 유용하다)
   B. ls -l 명령으로 재실행:

```
execlp("ls", "ls", "-l", (char*)NULL);   //기본 환경변수 적용됨.
//인자는 순서대로 ls파일, ls명령, -l옵션, 더 이상 인자가 없다는 의미로 NULL문자 넘김.
```

   C. exec함수군의 마지막 문자는 함수의 특징 설명함.

```
l: list로 명령과 옵션을 전달함. 서술이 끝났음을 NULL로 알려줘야 함.
v: vector로 명령과 옵션 전달. 서술이 끝났음을 알리지 않아도 됨.
p: 기본 환경변수가 지정되어 실행할 명령의 파일 위치를 전부 적지 않아도 됨.
e: 사용자 환경변수를 지정하여 넘겨줄 수 있다.
```

4. 3번과 유사한 문제임.
   A. execl("/usr/bin/pwd", "pwd", (char*)NULL);   //환경변수 사용 불가 -> 파일 경로 전부 입력.
5. 특정 자식 프로세스 종료 대기
   A. waitpid( )
6. 7장 끝.

■ 8장: 시그널

1. Ctrl + C : SIGINT
2. 시그널 집합 sigst에 SIGQUIT 추가 : sigaddset(&sigst, SIGQUIT);
   A. 시그널 집합의 주소를 넘겨준다는 것 주의.
3. 2번과 유사한 문제
   A. siggillset(&sigst); //모든 시그널을 집합에 채우기.
   B. sigemptyset(&sigst); //모든 시그널 비우기.
   C. sigdelset(&sigst, "시그널명"); //특정 시그널 제거.
4. 시스템에서 알람 시그널과 관련해 타이머를 제공한다.
   A. 리눅스 시스템은 세종류의 타이머를 제공한다.
      i. ITIMER_REAL : 실제 시간 사용. 타이머 만료시 SIGALRM 시그널 생성.
      ii. ITIMER_VIRTUAL: 프로세스의 사용자 모드 CPU시간 사용. 프로세스 동작중에만 시간이 작동함. 만료시 SIGVTALRM 시그널 생성.
      iii. ITIMER_PROF: 시스템 모드와 사용자 모드 CPU 시간을 합해서 사용. 타이머 만료시 SIGPROF 시그널 생성. ITIMER_VIRTUAL과 같이 사용시 프로세스의 사용자 모드 CPU 시간과 시스템 모드 CPU 시간을 모두 알 수 있음.
   B. 인터벌 타이머 기능을 제공하려면 두 함수를 사용한다. getitimer( )와 setitimer( )이다.

```
struct itimerval it;
it.it_interval.tv_sec = 2;  //2초 간격 인터벌 설정.
setitimer(ITIMER_REAL, &it, (struct itimerval *)NULL); //구조체 속 값에 따라 타이머 생성.
getitimer(ITIMER_REAL, &it);  //구조체변수 it에 ITIMER_REAL 타이머의 정보 저장.
```

5. 특정 시그널을 블로킹하거나 해제하는 함수.
   A. sighold(SIGINT) : 블로킹 / sigrelse(SIGINT) : 해제.
   B. sigprocmask() : 블록 & 해제. (리눅스에서 권장하는 블로킹 및 해제 함수)
      i. SIG_BLOCK : 시그널 집합을 시그널 마스크에 추가.
      ii. SIG_UNBLOCK : 시그널 집합을 시그널 마스크에서 제거.
      iii. SIG_SETMASK : 시그널 집합으로 현재 시그널 마스크 대체.

```
sigset_t new;
sigemptyset(&new);  //시그널 집합 비움.
sigaddset(&new, SIGINT);  //시크널 집합에 시그널 추가.
sigprocmask(SIG_BLOCK, &NEW, (sigset_t *)NULL);  //집합 new를 시그널 마스크에 추가.
//마스크에 추가하고 SIG_BLOCK을 했으므로, 추가된 시그널들이 블로킹 되었다는 뜻.

sigprocmask(SIG_UNBLOCK, &new, (sigset_t *)NULL);  //블로킹 해제됨.
```

6. 8장 끝.

■ 9장: 메모리 매핑

1. mmap( ) 함수로 메모리 매핑.
   A. mmap(NULL, statbuf.st_size, PROT_READ | PROT_WRITE, MAP_SHARED, fd, (off_t)0);
   B. mmap(매핑할 메모리 주소, 메모리 공간의 크기(byte단위), 보호 모드 지정(다수 지정 가능), 매핑된 데이터의 처리 방법 지정, 매핑할 파일의 파일 기술자, 매핑할 파일의 파일 오프셋);
   C. 이때 mmap( )이 리턴하는 값이 매핑된 주소이다. 그 주소에 쓰거나 주소로부터 읽을 수 있다.
2. 1번에서 MAP_SHARED 대신 MAP_PRIVATE 적용시, 리턴되는 값은 사본의 주소이다. (원본에 대하여 쓰기 금지. 데이터의 변경 내용을 공유하지 않음.)
3. munmap( )함수로 매핑을 해제한 메모리 영역에 접근하면, SIGSEGV 시그널이 발생한다.
4. mmap( )함수는 크기가 0인 파일을 매핑할 수 없다.
   A. truncate(int fd, off_t length)함수는 파일 크기를 키운다.
5. 매핑된 메모리의 보호 모드를 변경하는 함수는?
   A. mprotect( )
6. 프로세스간 통신이 무엇이며 왜 필요한가?
   A. 동일한 시스템 안에서 수행중인 프로세스끼리 데이터를 주고받는 것이 프로세스간 통신이다.
   B. 복잡한 프로그램의 내부적으로 동작하는 많은 프로세스들이 정상적으로 동작하는데 프로세스간 통신이 필수적이다.
7. 프로세스간 통신과 네트워크를 이용한 통신의 차이점을 설명하시오
   A. 프로세스간 통신은 하나의 시스템 안에서 이루어지는 통신. 파이프, 메모리 매핑 등의 방법으로 수행된다.
   B. 네트워크를 이용한 통신은 서로 다른 시스템에서 수행되는 프로세스 사이에서 이루어지는 통신이다. 주로 TCP/IP 프로토콜을 이용 가능하게끔 하는 소켓 인터페이스를 통해 수행된다.
8. 메모리 매핑을 어떻게 프로세스간 통신에서 활용할 수 있는가?
   A. 파일을 프로세스의 가상 주소 공간으로 매핑하는 것으로, 부모와 자식 프로세스간에 파일을 공유시키고, 이를 통해 통신이 가능하다.
9. 9장 끝.

■ 10: 파이프

1. 자식프로세스가 pwd 명령을 수행한 결과를 부모프로세스가 읽을 수 있도록 파이프를 생성하라.
   A. popen("pwd", "r")
   B. 항상 부모프로세스 기준으로 생각하라. w면 부모가 쓰고(출력), r이면 부모가 읽는다.
   C. 이 함수는 내부적으로 포크를 발생시키고, 명령은 자식프로세스가 실행한다.
2. Pipe(fd) 함수로 파이프를 생성했다. 이 파이프에서 데이터를 읽어오라.
   A. read(fd[0], buf, 256) //buf에 읽어온 내용을 저장.
   B. fd[0] : 읽기
   C. fd[1] : 쓰기
   D. 쓸 때: int fd1; write(fd1[1], "hello", 5); //쓰기만 할 파이프의 기술자 fd1의 인덱스 1로 출력.
3. Pipe(fd) 함수로 파이프를 생성했다. 부모 프로세스에서 자식 프로세스로 정보가 전달되도록 파일 기술자를 올바르게 정리한 것은?
   A. 부모 : close(fd[0]) //0을 닫아 쓰기만 함.
   B. 자식 : close(fd[1]) //1을 닫아 읽기만 함.
4. FIFO 파일 명령어로 만드는 방법 알아? 이름은 EXFIFO인 FIFO파일을 만들어봐.
   A. mknod EXFIFO p //다양한 특수 파일 생성하는 명령임. 마지막 인자로 p를 입력시 FIFO 생성.
   B. mkfifo -m 0644 EXFIFO //fifo파일을 만드는 명령임. -m 옵션으로 권한 지정.
   C. mkfifo EXFIFO //-m 생략시 기본 권한으로 생성됨.
5. mknod( ) 함수로 FIFO파일 만들 줄 알아? 이름은 EXFIFO, 권한은 0644로 만들어봐.
   A. mknod( )는 다양한 특수 파일을 만들 수 있는 함수이다.
   B. mknod("EXFIFO", S_IFIFO | 0644, 0) //인자: 파일 이름, 파일 종류 | 권한, 장치 설정값.
   C. 다른 함수도 있다.
   D. FIFO 생성 함수로 생성 예시: mkfifo("EXFIFO", 0644)
6. 10장 끝.

■ 11: 프로세스간 통신

1. 키 생성: ftok("키 생성용 파일명(혹은 경로명)", 정수)
   A. 예: ftok("./key", 1)
2. ipcrm 명령 알아?
   A. ipcrm -a : 모든 자원 제거.
   B. ipcrm -M shmkey : shmkey로 지정한 공유 메모리 삭제.
   C. ipcrm -m shmid : shmid로 지정한 공유 메모리 삭제.
   D. ipcrm -Q msgkey : msgkey로 지정한 메시지 큐 삭제.
   E. ipcrm -q msgid : msgid로 지정한 메시지 큐 삭제.
   F. ipcrm -S semkey : semkey로 지정한 세마포어 삭제.
   G. ipcrm -s semid : semid로 지정한 세마포어 삭제.
   H. 규칙이 보이리라 믿는다. Ipcrm + 옵션 + 옵션명 약자(3자) + 키 혹은 식별자(id)
3. msgget( )함수로 메시지 큐 식별자 생성하라.
   A. msgget(key, IPC_CREAT | 0664)
4. 공유메모리 연결 알아?
   A. 공유메모리 식별자 생성: shmget( )
   B. 공유메모리 연결: shmat( )
   C. 공유메모리 연결 해제: shmdt( )
5. 세마포어를 잠그기 위한 sembuf 구조체의 sem_op값으로 올바른 것은?
   A. sem_op > 0 : 열림
   B. sem_op < 0 : 잠김
   C. sem_op == 0 : 즉시 (semop( )함수가) 리턴된다.
6. ipcmk 명령 알아?
   A. Ipcmk -M size : size에 지정한 바이트 크기로 공유 메모리 생성.
   B. Ipcmk -Q : 메시지 큐 생성.
   C. Ipcmk -S number : number에 지정한 개수의 요소를 갖는 세마포어 생성.
   D. Ipcmk -p mode : 접근 권한 지정.
7. 11장 끝.

■ 12: 소켓 프로그래밍 기초

1. 호스트 정보를 읽어오는 함수 알아?
   A. gethostent( ) : 호스트명과 ip를 읽어온다.
   B. gethostbyname( ) : ip 주소로 호스트 정보를 검색하여 읽어온다.
   C. gethostbyaddr( ) : 호스트명으로 정보를 검색하여 읽어온다.
   D. 이 함수들은 구조체에 읽어온 정보를 저장한다.
2. 빅 엔디언으로 변환하는 함수 : htonl( ) / htons( )
   A. Host to network long int / host to network short int
   B. 통신 표준은 빅 엔디언이고, 인텔CPU는 리틀 엔디언이다.
   C. 빅 엔디언이 네트워크고, NBO라고도 한다. (network bite order)
3. 문자열 형태의 ip 주소를 숫자 형태로 변환
   A. inet_addr("192.168.0.1")
   B. 반대로 하려면, inet_ntoa(const struct in_addr in) 사용. (구조체 숫자를 문자열로 변환)
4. 소켓 파일 기술자를 지정된 ip주소와 연결하는 함수는?
   A. bind( )
5. 소켓을 사용해 통신을 하려고 한다.
   A. 그냥 정리하겠다.
   B. 서버 측 : soket( ) – bind( ) – listen( ) – accept( ) - 송수신 – close( )
   C. 클라이언트 측: soket( ) - connect( ) – 송수신 – close( )
6. TCP와 UDP의 차이점은 무엇이고 용도는 어떻게 구별?
   A. TCP: 통신의 신뢰성 보장. 속도 변동 가능. 일반적으로는 대부분 TCP.
   B. UDP: 상대측의 수신 여부 확인 안함. 빠름. 방송에 많이 씀.
7. TCP/IP로 통신을 할 때 IP주소 외에 포트 번호가 필요한 이유는?
   A. 전송되어오는 데이터를 어떤 프로세스가 수신하여 서비스를 제공해야 할지 지정하기 위하여 포트 번호가 필요함.
8. TCP/IP로 통신을 할 때 바이트 순서에 주의해야 하는 이유는?
   A. 인텔 계열 CPU가 그렇듯이, 네트워크 통신의 표준에 맞지 않는 바이트 순서를 사용하는 시스템이 있을 수 있기 때문이다. 정상적인 통신을 위해서는 바이트 순서를 확인하고 전환해야 한다.
9. 호스트 이름과 IP주소를 모두 사용하는 이유는?
   A. 인간이 IP 주소보다는 호스트 이름을 잘 이해하고 인식할 수 있기 때문.
10. Accept( )함수가 클라이언트의 접속을 허용할 때 새로운 소켓을 리턴하는 이유는?
    A. Accept( )에 넘겨준 소켓으로 추가 연결 요청이 들어오는 것을 감지하기 위하여, 새로운 소켓을 생성하여 통신하는 것으로, 기존 소켓을 남겨둬야 하기 때문.
11. 12장 끝.