Cykor2-System Hacking 2차

스마트보안학과 남유찬 2024350022

우선, "username@hostname$” 이러한 형식을 잡는 것부터 구현하려 했습니다.

username과 hostname을 가져와서 합쳤습니다.

```
// "username@hostname"
char *userhost(void){
    static char buf[256];
    char hostname[64];

    uid_t uid = getuid();
    struct passwd *pw = getpwuid(uid);

    // 정상적으로 user와 host를 가져오면 username@hostname을 return.
    if ((pw) && (gethostname(hostname, sizeof(hostname))) == 0) {
        snprintf(buf, sizeof(buf), "%s@%s", pw→pw_name, hostname);
        return buf;
    }
    else {
        printf("사용자 정보 로드 실패.");
    }
    return NULL;
}
```

user id를 가져와서 pw→pw_name에서 username을 찾고, gethostname을 통해 hostname을 얻어왔습니다.

또한 다음과 같이 현재 directory를 표현해줍니다. 자세한 부분은 주석으로 처리하였습니다.

```
// current directory 구현
char *curdir(void){
    static char buf[256];
```

```
    char cwd[256];

    // cwd에 현재 directory 저장. (ex. /home/username/...)
    if (getcwd(cwd, sizeof(cwd)) == NULL) {
        return NULL;    //실패
    }

    // home dir 경로 저장. (ex. /home/username )
    struct passwd *pw = getpwuid(getuid());
    if (!pw) {
        return NULL;    //실패
    }
    const char *home = pw→pw_dir;
    size_t hl = strlen(home); // home 경로 길이

    // cwd가 /home/username으로 시작하는지
    if (strncmp(cwd, home, hl) == 0) {
        if (cwd[hl] == '/' || cwd[hl] == '\0') {
            // "/home/username"인 경우
            if (cwd[hl] == '\0') {
                snprintf(buf, sizeof(buf), "~"); // ex) ~
            }
            // "/home/username/..." 인 경우
            else {
                snprintf(buf, sizeof(buf), "~%s", cwd + hl); // ex) ~/...
            }
            return buf;
        }
    }

    // /home/username으로 시작하지 않으면 그대로 저장.
    snprintf(buf, sizeof(buf), "%s", cwd);
    return buf;

}
```

위의 둘을 합친 결과는

username@hostname:~/projects/Cykor2$

이런 식으로 나옵니다.

이것을 그냥 출력하면 너무 심심하니

```
// 글자 굵기, 색
#define Bold_RED    "\033[1;31m"
#define Bold_GREEN  "\033[1;32m"
#define Bold_BLUE   "\033[1;34m"
#define RESET       "\033[0m"
```

```
char user[256];
snprintf(user, sizeof(user), Bold_GREEN "%s" RESET":", userhost());
```

```
char cwd[256];
snprintf(cwd, sizeof(cwd), Bold_RED "%s" RESET"$", curdir());
```

이렇게 색을 입혀줍니다.

```
// 기본 출력 ex) "username@hostname:~/projects$ "
char base[512];
snprintf(base, sizeof(base), "%s%s ", user, cwd);
```

이렇게 base로 기본 출력을 지정해 둡니다.

username@hostname:~/projects/Cykor2$

위와 같이 출력됩니다. 기본 bash와 구분하기 위해 빨간색으로 표현하였습니다.

입력을 다음과 같이 받고, 처리합니다. 입력을 받아 add_history로 입력을 저장합니다.

기본적으로는 split_lumps() 함수를 통해 입력을 덩어리로 분리합니다.

"|" "||" "&" "&&" ";" 이 다섯가지가 없으면 바로 명령어를 실행하고, 다섯가지 중에 하나라도 있으면 for문으로 들어가 덩어리를 옮겨서 반복합니다.

```
// 덩어리로 분리 + 기호 분리(|| && 등)
void split_lumps(const char *line) {
    const char *p     = line;    // 순회용 포인터
```

```
    const char *start = p;       // 시작 지점
    nl = nc = 0;

    while (*p) {
        if ((p[0]=='|' && p[1]=='|') || (p[0]=='&' && p[1]=='&')) {     // || 또는 &&
            if (p > start)
                lumps[nl++] = strndup(start, p - start);
            crits [nc++] = strndup(p, 2);

            p     += 2;
            start = p;
        }
        else if (*p=='|' || *p=='&' || *p==';') {                    // | & ;
            if (p > start)
                lumps[nl++] = strndup(start, p - start);
            crits [nc++] = strndup(p, 1);

            p     += 1;
            start = p;
        }
        else {
            p++;
        }
    }

    if (p > start)
        lumps[nl++] = strndup(start, p - start);

    lumps[nl] = NULL;
    crits [nc] = NULL;
}
```

```
        if (!line) break;
        if (*line) add_history(line);
        else {          // 엔터만 쳤을 경우.
            free(line);
            continue;
        }
```

```
        split_lumps(line);
        free(line);

        int last_status = 0;    // 이전 명령 실패(!0) / 성공(0)
        if (nc > 0) {           // 조건 연산자? ("|" "||" "&" "&&" ";")가 있으면.
           for (int i = 0; i < nl; i++) {      // lump 개수 번 반복.
              if (i>0 && strcmp(crits[i-1], "&&")==0) {          // &&: 이전 명령 실패 -> 
                 if (last_status != 0) continue;
              } else if (i>0 && strcmp(crits[i-1], "||")==0) {    // ||: 이전 명령 성공 -> 
                 if (last_status == 0) continue;
              }
              if (i < nc && strcmp(crits[i], "|")==0) {
                 int start = i;
                 int end = i;
                 while (end < nc && strcmp(crits[end], "|")==0) {
                    end++;
                 }
                 last_status = run_pipeline(lumps, start, end);
                 i = end + 1;       // 다음 명령어로 건너뛰기
                 continue;
              }
              if (i < nc && strcmp(crits[i], "&") == 0) {
                 run_cmd_bg(lumps[i]);
                 last_status = 0;
                 continue;
              } else {
                 last_status = run_cmd(lumps[i]);
              }
              argc = 0;
           }
           for (int i=0;i<nl;i++) free(lumps[i]);
           for (int i=0;i<nc;i++) free(crits[i]);

        }
        else {                  //("|" "||" "&" "&&" ";")가 없으면.
           run_cmd(*lumps);
        }
        argc = 0;
```

cd 명령어는 chdir 명령어를 사용했고,

pwd 명령어는 getcwd 명령어를 사용하였습니다.

나머지 명령어들은 기본적으로 exec 시스템콜을 사용했고, 사고의 자세한 부분이나 메모, 수정 전 코드 등은 주석으로 적어놓았습니다.