A puzzle...

```
struct sockaddr_un {
    /* May be other fields here, e.g., sun_len on BSDs */
    sa_family_t sun_family; /* AF_UNIX */
    char sun_path[UNIX_PATH_MAX]; /* pathname */
};
bind(fd, (struct sockaddr *) &addr, addrlen);
```

- UNIX domain sockets are bound to address of above form; `sun_path` is always final member of structure
- `addrlen` includes all bytes to end of `sun_path`
  - Commonly: `sizeof(struct sockaddr_un)`
- `UNIX_PATH_MAX` varies by system
  - e.g., 92 on HP-UX, 108 on Linux
A puzzle...

```c
struct sockaddr_un {
    /* May be other fields here, e.g., sun_len on BSDs */
    sa_family_t sun_family;           /* AF_UNIX */
    char sun_path[UNIX_PATH_MAX];     /* pathname */
};

bind(fd, (struct sockaddr *) &addr, addrlen);
```

- Caller **may** include '\0' at end of `sun_path` (passed to kernel), but is not required to

- Kernel **may** add '\0' at end of `sun_path` (if returning it to user space and buffer is large enough), but is not required to

  - (Linux does so, even when pathname is `UNIX_PATH_MAX` bytes long)
A puzzle...

```c
struct sockaddr_un {
    /* May be other fields here, e.g., sun_len on BSDs */
    sa_family_t sun_family; /* AF_UNIX */
    char sun_path[UNIX_PATH_MAX]; /* pathname */
};

/* Initialize addrlen to size of addr buffer */
accept(fd, (struct sockaddr *) &addr, &addrlen);
```

- Some system calls return a `sockaddr_un`: e.g., `accept()`
  - On call, `addrlen` indicates space available in `addr` structure
  - On return, `addrlen` gives number of bytes returned in `addr`
A puzzle...

After `accept()` , how can you use (e.g., print) `sun_path` safely (don't run past end!) and portably?
How many applications get this right?

/* Solution 1 */
struct sockaddr_un addr;
addrlen = sizeof(struct sockaddr_un);
cfd = accept(fd, (struct sockaddr *) &addr, &addrlen);
printf("%.%*s",
    addrlen - offsetof(struct sockaddr_un, sun_path),
    addr.sun_path);

/* Solution 2 (assumes sun_path is at end of struct) */
/* Ensures null terminator even if sun_path is filled */
addrlen = sizeof(struct sockaddr_un);
void *addrp = malloc(addrlen + 1);
memset(addrp, 0, addrlen + 1);
accept(fd, (struct sockaddr *) addrp, &addrlen);
printf("sun_path = %s\n",
    ((struct sockaddr_un *) addrp)->sun_path);
The real problem

- If *only* original kernel (and POSIX spec) had **required** that user must include null byte at end of `sun_path`...
- There is no “fix” to this problem
  - Changing kernel could break existing applications...
- Present since 1983 (invention of BSD sockets)