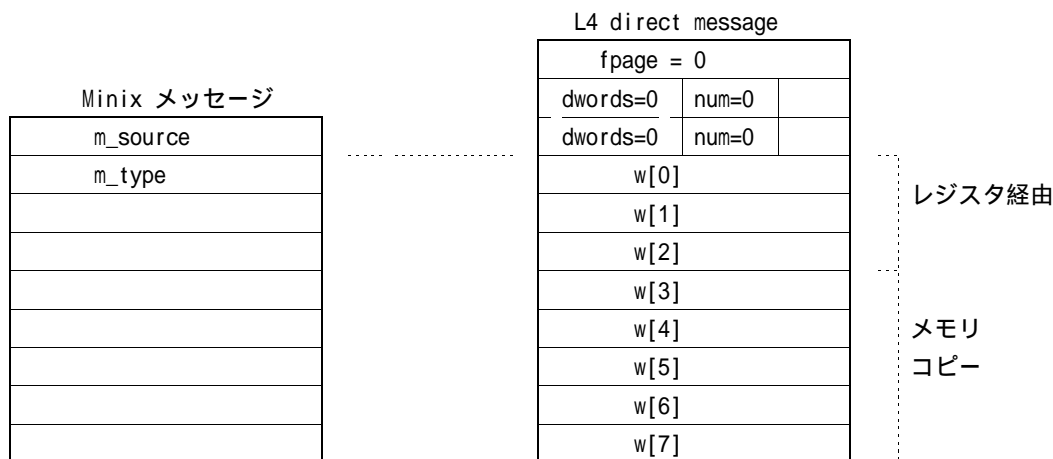


## § 1. Minixメッセージ

Minixの message は、L4 の direct message に乗せる。

Cf. katsura:/home/H20/l4-ka/minix/src/L4/L4-if.c

メッセージの宛て先 送り主の指定には L4 の l4\_threadid\_t.raw を使う



## (1) メッセージ送信

```
int _send(unsigned to,      //メッセージの宛て先 (L4_threadid_t)
          int  msg[]);    // Minixメッセージのアドレス
```

## (2) メッセージ受信

```
int _receive(unsigned from, //メッセージの送り主 (L4_threadid_t)
              int  msg[]);  // Minixメッセージのアドレス
```

from = 0 を指定すると、Open wait、つまり誰からもメッセージを受け取る

## (2) メッセージ送受信

```
int _sendrec(unsigned to_from, //メッセージの宛て先 (L4_threadid_t)
              int  msg[]);    // Minixメッセージのアドレス
```

to\_from で指定したスレッドにメッセージ msg[] を送り、かつ返答を mag[]に受け取る。

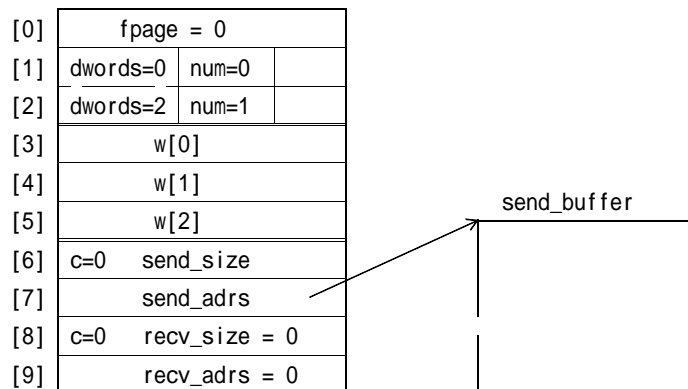
## § 2. バッファ転送

### § 2.1 Scatter転送

一つのバッファから複数のバッファに Scatter 型コピーを行う。

#### 【送信側】

```
int sndmsg[10];
for(int i=0; i<10; i++) sndmsg[i]=0;
sndmsg[2] = 2 << 13 | 1 << 8;
sndmsg[6] = バッファサイズ;
sndmsg[7] = バッファアドレス;
l4_ipc_send(to,
            sndmsg, w0, w1, w2,
            L4_IPC_NEVER, &rcode);
```



L4-if.c を使えば、次のように書くことができる。

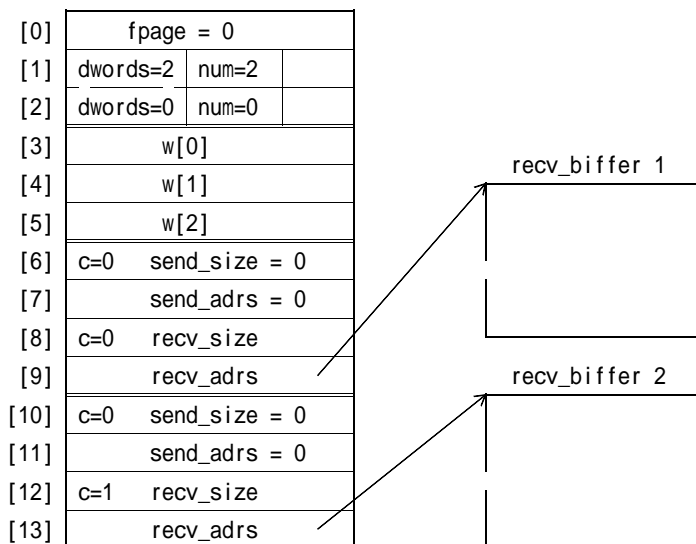
```
int recmsg[10];
for(int i=0; i<10; i++) recmsg[i]=0;
set_send_dope(recmsg, 0, バッファアドレス, バッファサイズ, 0);
_send_indirect(0, w0, w1, w2, sndmsg, 1);
```

#### 【受信側】

```
int recmsg[14];
for(int i=0; i<14; i++) recmsg[i]=0;
sndmsg[1] = 2 << 13 | 2 << 8;
sndmsg[8] = バッファ1 サイズ;
sndmsg[9] = バッファ1 アドレス;

sndmsg[12] = バッファ2 サイズ | 0x80000000;
sndmsg[13] = バッファ2 アドレス;

l4_ipc_wait(&from,
            recmsg, &x, &y, &z,
            L4_IPC_NEVER, &rcode);
```



L4-if.c を使えば、以下のように書ける。

```
int recmsg[14];
for(int i=0; i<14; i++) recmsg[i]=0;
set_recv_dope(recmsg, 0, バッファ1 アドレス, バッファ1 サイズ, 0);
set_recv_dope(recmsg, 0, バッファ2 アドレス, バッファ2 サイズ, 1);
from = _receive_indirect(0, &x, &y, &z, recmsg, 2);
```

## § 2.2 Gather 転送

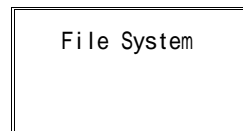
送信側で複数バッファの continuation を行う。Scatterコピーと対照のプログラムになる。

## § 3. APL - Service Task 間の通信

APL とService Task 間の通信には、  
バッファ転送を必要としない。  
バッファ転送を必要とする。  
に分類される。



バッファ転送を必要としない場合は、Minixメッセージを  
\_send( ), \_receive( ) するだけでよい。



バッファ転送を必要とする場合は、 §2 のバッファ転送を使う必要がある。

APL のソースプログラムでは、システムコールは一般に普通のサブルーチンコール形式である。  
ライブラリルーチンがこれをメッセージ通信に直す。

### 【ファイルサーバーの例】

(1) fd = open( filename, flagfs);

要求メッセージ編集;  
\_sendrec( FILESYSTEM, メッセージ);  
返答メッセージの処理;

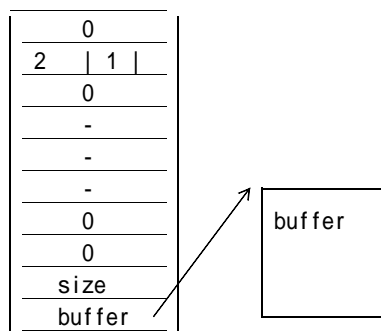
(2) n = write(fd, buffer, size);

サイズによっては、ユーザ空間の bufferからファイルサーバー中の複数のバッファークャッシュに  
Scatterコピーする必要がある。

【 APL 】

【 FileServer 】

sndmsg を設定する。



要求メッセージを送る

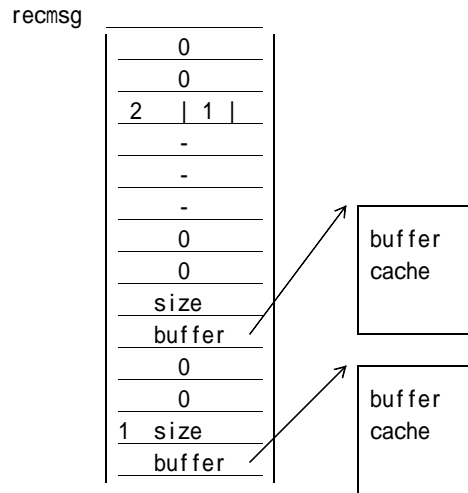
```
l4_ipc_call(FileServer,  
            0, // RegMessage  
            WRITE, fd, size,  
            0, &x, &y, &z,  
            ...);
```

要求メッセージを受ける

```
l4_ipc_wait( &client,  
            0. // RegMessage  
            &op, &fd, &size,  
            ,,,, );
```

client と fd から目的ファイルを判別し、  
position から始まる size 分のデータを入れる  
バッファキャッシュ (1 ..n 個) を用意する。

recmsg を設定する。



client に送信要求を送る。

```
l4_ipc_call(client, 0, a, b, c,  
            recmsg, &x, &y, &z,  
            ,,,);
```

File Server の送信要求を受けて、  
バッファ転送を行う。

```
l4_ipc_receive(FileServer,  
              0, &a, &b, &c, ,,,);
```

```
l4_ipc_receive(FileServer,  
              sndmsg, d, e, f,  
              0, &n, &m, &l ,,,);
```

完了メッセージを送る。

```
l4_ipc_send(client, ,,,);
```

(3) n = read(fd, buffer, size);

readの場合は、メッセージは一回の往復で済む。

## § 4. Service Task - Driver間の通信

Minixでは、driver はカーネルないであり、かつユーザタスクないの論理アドレスから物理アドレスにすぐ変換できるので、ドライバでは物理アドレスをすべてのタスクを処理できる。

L4-Minix では、効率よく実現するには、ユーザの論理ページを一時的に driver taskにマッピングするような Pager が望まれる。