
Subject: [PATCH net-2.6.25] Name magic constants in sock_wake_async()

Posted by [Pavel Emelianov](#) on Fri, 23 Nov 2007 13:43:11 GMT

[View Forum Message](#) <> [Reply to Message](#)

The sock_wake_async() performs a bit different actions depending on "how" argument. Unfortunately this argument only has numerical magic values.

I propose to give names to their constants to help people reading this function callers understand what's going on without looking into this function all the time.

I suppose this is 2.6.25 material, but if it's not (or the naming seems poor/bad/awful), I can rework it against the current net-2.6 tree.

Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

```
diff --git a/include/linux/net.h b/include/linux/net.h
index 0235d91..f95f12c 100644
--- a/include/linux/net.h
+++ b/include/linux/net.h
@@ -186,6 +186,13 @@ struct net_proto_family {
    struct iovec;
    struct kvec;

+enum {
+    SOCK_WAKE_IO,
+    SOCK_WAKE_WAITD,
+    SOCK_WAKE_SPACE,
+    SOCK_WAKE_URG,
+};
+
extern int    sock_wake_async(struct socket *sk, int how, int band);
extern int    sock_register(const struct net_proto_family *fam);
extern void    sock_unregister(int family);
diff --git a/net/atm/common.c b/net/atm/common.c
index eba09a0..c865517 100644
--- a/net/atm/common.c
+++ b/net/atm/common.c
@@ -113,7 +113,7 @@ static void vcc_write_space(struct sock *sk)
    if (sk->sk_sleep && waitqueue_active(sk->sk_sleep))
        wake_up_interruptible(sk->sk_sleep);

-    sock_wake_async(sk, 2, POLL_OUT);
+    sock_wake_async(sk, SOCK_WAKE_SPACE, POLL_OUT);
```

```

}

read_unlock(&sk->sk_callback_lock);
diff --git a/net/core/sock.c b/net/core/sock.c
index eac7aa0..1182140 100644
--- a/net/core/sock.c
+++ b/net/core/sock.c
@@ -1498,7 +1498,7 @@ static void sock_def_error_report(struct sock *sk)
    read_lock(&sk->sk_callback_lock);
    if (sk->sk_sleep && waitqueue_active(sk->sk_sleep))
        wake_up_interruptible(sk->sk_sleep);
- sk_wake_async(sk,0,POLL_ERR);
+ sk_wake_async(sk, SOCK_WAKE_IO, POLL_ERR);
    read_unlock(&sk->sk_callback_lock);
}

@@ -1507,7 +1507,7 @@ static void sock_def_readable(struct sock *sk, int len)
    read_lock(&sk->sk_callback_lock);
    if (sk->sk_sleep && waitqueue_active(sk->sk_sleep))
        wake_up_interruptible(sk->sk_sleep);
- sk_wake_async(sk,1,POLL_IN);
+ sk_wake_async(sk, SOCK_WAKE_WAITD, POLL_IN);
    read_unlock(&sk->sk_callback_lock);
}

@@ -1524,7 +1524,7 @@ static void sock_def_write_space(struct sock *sk)

    /* Should agree with poll, otherwise some programs break */
    if (sock_writeable(sk))
- sk_wake_async(sk, 2, POLL_OUT);
+ sk_wake_async(sk, SOCK_WAKE_SPACE, POLL_OUT);
}

read_unlock(&sk->sk_callback_lock);
@@ -1539,7 +1539,7 @@ void sk_send_sigurg(struct sock *sk)
{
    if (sk->sk_socket && sk->sk_socket->file)
        if (send_sigurg(&sk->sk_socket->file->f_owner))
- sk_wake_async(sk, 3, POLL_PRI);
+ sk_wake_async(sk, SOCK_WAKE_URG, POLL_PRI);
}

void sk_reset_timer(struct sock *sk, struct timer_list* timer,
diff --git a/net/core/stream.c b/net/core/stream.c
index b2fb846..5586879 100644
--- a/net/core/stream.c
+++ b/net/core/stream.c
@@ -35,7 +35,7 @@ void sk_stream_write_space(struct sock *sk)

```

```

    if (sk->sk_sleep && waitqueue_active(sk->sk_sleep))
        wake_up_interruptible(sk->sk_sleep);
    if (sock->fasync_list && !(sk->sk_shutdown & SEND_SHUTDOWN))
-   sock_wake_async(sock, 2, POLL_OUT);
+   sock_wake_async(sock, SOCK_WAKE_SPACE, POLL_OUT);
}
}

```

diff --git a/net/dccp/input.c b/net/dccp/input.c

index 1ce1010..11bf47e 100644

--- a/net/dccp/input.c

+++ b/net/dccp/input.c

```

@@ -37,7 +37,7 @@ static void dccp_rcv_close(struct sock *sk, struct sk_buff *skb)
    dccp_send_reset(sk, DCCP_RESET_CODE_CLOSED);
    dccp_fin(sk, skb);
    dccp_set_state(sk, DCCP_CLOSED);
-   sk_wake_async(sk, 1, POLL_HUP);
+   sk_wake_async(sk, SOCK_WAKE_WAITD, POLL_HUP);
}

```

```

static void dccp_rcv_clasreq(struct sock *sk, struct sk_buff *skb)

```

```

@@ -90,7 +90,7 @@ static void dccp_rcv_reset(struct sock *sk, struct sk_buff *skb)
    dccp_fin(sk, skb);

```

```

    if (err && !sock_flag(sk, SOCK_DEAD))
-   sk_wake_async(sk, 0, POLL_ERR);
+   sk_wake_async(sk, SOCK_WAKE_IO, POLL_ERR);
    dccp_time_wait(sk, DCCP_TIME_WAIT, 0);
}

```

```

@@ -402,7 +402,7 @@ static int dccp_rcv_request_sent_state_process(struct sock *sk,

```

```

    if (!sock_flag(sk, SOCK_DEAD)) {
        sk->sk_state_change(sk);
-   sk_wake_async(sk, 0, POLL_OUT);
+   sk_wake_async(sk, SOCK_WAKE_IO, POLL_OUT);
    }

```

```

    if (sk->sk_write_pending || icsk->icsk_ack.pingpong ||
@@ -611,7 +611,7 @@ int dccp_rcv_state_process(struct sock *sk, struct sk_buff *skb,
    switch (old_state) {
        case DCCP_PARTOPEN:
            sk->sk_state_change(sk);
-   sk_wake_async(sk, 0, POLL_OUT);
+   sk_wake_async(sk, SOCK_WAKE_IO, POLL_OUT);
            break;
    }
} else if (unlikely(dh->dccph_type == DCCP_PKT_SYNC)) {

```

```

diff --git a/net/dccp/output.c b/net/dccp/output.c
index f495446..33ce737 100644
--- a/net/dccp/output.c
+++ b/net/dccp/output.c
@@ -170,7 +170,7 @@ void dccp_write_space(struct sock *sk)
    wake_up_interruptible(sk->sk_sleep);
    /* Should agree with poll, otherwise some programs break */
    if (sock_writeable(sk))
-   sk_wake_async(sk, 2, POLL_OUT);
+   sk_wake_async(sk, SOCK_WAKE_SPACE, POLL_OUT);

    read_unlock(&sk->sk_callback_lock);
}
diff --git a/net/ipv4/tcp_input.c b/net/ipv4/tcp_input.c
index dae000b..0cee3dc 100644
--- a/net/ipv4/tcp_input.c
+++ b/net/ipv4/tcp_input.c
@@ -3597,9 +3597,9 @@ static void tcp_fin(struct sk_buff *skb, struct sock *sk, struct tcphdr *th)
    /* Do not send POLL_HUP for half duplex close. */
    if (sk->sk_shutdown == SHUTDOWN_MASK ||
        sk->sk_state == TCP_CLOSE)
-   sk_wake_async(sk, 1, POLL_HUP);
+   sk_wake_async(sk, SOCK_WAKE_WAITD, POLL_HUP);
    else
-   sk_wake_async(sk, 1, POLL_IN);
+   sk_wake_async(sk, SOCK_WAKE_WAITD, POLL_IN);
}
}

@@ -4958,7 +4958,7 @@ static int tcp_rcv_synsent_state_process(struct sock *sk, struct sk_buff
*skb,

    if (!sock_flag(sk, SOCK_DEAD)) {
        sk->sk_state_change(sk);
-       sk_wake_async(sk, 0, POLL_OUT);
+       sk_wake_async(sk, SOCK_WAKE_IO, POLL_OUT);
    }

    if (sk->sk_write_pending ||
@@ -5188,9 +5188,9 @@ int tcp_rcv_state_process(struct sock *sk, struct sk_buff *skb,
    * are not waked up, because sk->sk_sleep ==
    * NULL and sk->sk_socket == NULL.
    */
-   if (sk->sk_socket) {
-       sk_wake_async(sk, 0, POLL_OUT);
-   }
+   if (sk->sk_socket)
+       sk_wake_async(sk,

```

```

+   SOCK_WAKE_IO, POLL_OUT);

    tp->snd_una = TCP_SKB_CB(skb)->ack_seq;
    tp->snd_wnd = ntohs(th->window) <<
diff --git a/net/rxrpc/af_rxrpc.c b/net/rxrpc/af_rxrpc.c
index d638945..5e82f1c 100644
--- a/net/rxrpc/af_rxrpc.c
+++ b/net/rxrpc/af_rxrpc.c
@@ -65,7 +65,7 @@ static void rxrpc_write_space(struct sock *sk)
    if (rxrpc_writable(sk)) {
        if (sk->sk_sleep && waitqueue_active(sk->sk_sleep))
            wake_up_interruptible(sk->sk_sleep);
-   sk_wake_async(sk, 2, POLL_OUT);
+   sk_wake_async(sk, SOCK_WAKE_SPACE, POLL_OUT);
    }
    read_unlock(&sk->sk_callback_lock);
}
diff --git a/net/sctp/socket.c b/net/sctp/socket.c
index ff8bc95..248b9a5 100644
--- a/net/sctp/socket.c
+++ b/net/sctp/socket.c
@@ -6008,7 +6008,8 @@ static void __sctp_write_space(struct sctp_association *asoc)
    */
    if (sock->fasync_list &&
        !(sk->sk_shutdown & SEND_SHUTDOWN))
-   sock_wake_async(sock, 2, POLL_OUT);
+   sock_wake_async(sock,
+   SOCK_WAKE_SPACE, POLL_OUT);
    }
}
diff --git a/net/socket.c b/net/socket.c
index aeeab38..9ebca5c 100644
--- a/net/socket.c
+++ b/net/socket.c
@@ -1070,20 +1070,19 @@ int sock_wake_async(struct socket *sock, int how, int band)
    if (!sock || !sock->fasync_list)
        return -1;
    switch (how) {
-   case 1:
-   -
+   case SOCK_WAKE_WAITD:
        if (test_bit(SOCK_ASYNC_WAITDATA, &sock->flags))
            break;
        goto call_kill;
-   case 2:
+   case SOCK_WAKE_SPACE:
        if (!test_and_clear_bit(SOCK_ASYNC_NOSPACE, &sock->flags))

```

```

    break;
    /* fall through */
- case 0:
+ case SOCK_WAKE_IO:
    call_kill:
    __kill_fasync(sock->fasync_list, SIGIO, band);
    break;
- case 3:
+ case SOCK_WAKE_URG:
    __kill_fasync(sock->fasync_list, SIGURG, band);
}
return 0;
diff --git a/net/unix/af_unix.c b/net/unix/af_unix.c
index 6be6d87..f8ad367 100644
--- a/net/unix/af_unix.c
+++ b/net/unix/af_unix.c
@@ -343,7 +343,7 @@ static void unix_write_space(struct sock *sk)
    if (unix_writable(sk)) {
        if (sk->sk_sleep && waitqueue_active(sk->sk_sleep))
            wake_up_interruptible_sync(sk->sk_sleep);
-    sk_wake_async(sk, 2, POLL_OUT);
+    sk_wake_async(sk, SOCK_WAKE_SPACE, POLL_OUT);
    }
    read_unlock(&sk->sk_callback_lock);
}
@@ -429,7 +429,7 @@ static int unix_release_sock (struct sock *sk, int embrion)
    unix_state_unlock(skpair);
    skpair->sk_state_change(skpair);
    read_lock(&skpair->sk_callback_lock);
-    sk_wake_async(skpair,1,POLL_HUP);
+    sk_wake_async(skpair, SOCK_WAKE_WAITD, POLL_HUP);
    read_unlock(&skpair->sk_callback_lock);
}
sock_put(skpair); /* It may now die */
@@ -1919,9 +1919,9 @@ static int unix_shutdown(struct socket *sock, int mode)
    other->sk_state_change(other);
    read_lock(&other->sk_callback_lock);
    if (peer_mode == SHUTDOWN_MASK)
-    sk_wake_async(other,1,POLL_HUP);
+    sk_wake_async(other, SOCK_WAKE_WAITD, POLL_HUP);
    else if (peer_mode & RCV_SHUTDOWN)
-    sk_wake_async(other,1,POLL_IN);
+    sk_wake_async(other, SOCK_WAKE_WAITD, POLL_IN);
    read_unlock(&other->sk_callback_lock);
}
if (other)
--

```