Threads and Shared Memory

\[
x = x + 1; \\
x = x + 2;
\]
Threads and Shared Memory

threads $t_1$ and $t_2$

$x = x + 1;$

$x = x + 2;$

0
Threads and Shared Memory

$t_1$
lock();
x = x + 1;
unlock();

$t_2$
lock();
x = x + 2;
unlock();
Threads and Shared Memory

lock();
x = x + 1;
unlock();

lock();
...x...
unlock();
Message Passing

```
while (1) {
    ...  
    x = x + v;  
    ...  
}
```

```
t_1
send(s, 2)
```

```
t_2
recv(s)
```
Message Passing

\[t_1\]

\textit{send}(s, 2)

\[t_2\]

\textit{recv}(s)

\texttt{while (1) \{}
\texttt{\ldots}
\texttt{x = x + v,}
\texttt{\ldots}
\texttt{\}}
Message Passing

```
while (1) {
    ... 
    x = x + v;
    ... 
}
```

send(s, 2)

recv(s)

\( t_1 \)

\( t_2 \)
Message Passing

\[
\text{while (1) \{ }
\text{\hspace{1em} \ldots}
\text{\hspace{1em} } x = x + v;
\text{\hspace{1em} \ldots}
\text{\hspace{1em} }
\text{\}}
\]

send(s, 2)

\[t_1\]

recv(s)

\[t_2\]
Message Passing in Racket

see cml

make-channel : -> channel-of-X

channel-put : channel-of-X X -> void

channel-get : channel-of-X -> X

sync : evt-of-X ... -> X

; An evt-of-X is either
; - channel-of-X
; ... 

handle-evt : evt-of-X (X -> Y) -> evt-of-Y

never-evt : evt-of-X

channel-put-evt : channel-of-X X -> evt-of-void
Message Passing

\[
\text{while (1) { } }
\]
\[
\text{... }
\]
\[
\text{x = x + v, }
\]
\[
\text{... }
\]
\[
\text{}}
\]

send(s, 2)

recv(s)

\[t_1\]

\[t_2\]
Distributed Message Passing

\[
\text{send}(s, 2) \\
\text{while (1) } \{ \\
\ldots \\
x = x + v; \\
\ldots \\
\} \\
\text{recv}(s) \\
\]
Places in Racket

(define \texttt{parent-channel-name}
  (place \texttt{child-channel-name}
    \texttt{expression}))

Creates a new “copy” of Racket to evaluate \texttt{expression}

Messages sent to \texttt{parent-channel-name} can be received from \texttt{child-channel-name} and vice versa

\texttt{place-channel-get} : \texttt{place-channel-of-X} \to X
\texttt{place-channel-put} : \texttt{place-channel-of-X} X \to \texttt{void}

see field/place-player.rkt
**Message Passing Interface** (or **MPI**) is a message-passing library for many languages

```c
int MPI_Send(void *buf, int count,
             MPI_Datatype datatype,
             int dest, int tag,
             MPI_Comm comm);
```

```c
int MPI_Recv(void *buf, int count,
              MPI_Datatype datatype,
              int source, int tag,
              MPI_Comm comm, MPI_Status *status);
```