(1) Ordinary Turing machine tape:

initially: all 0's ("blank")

<table>
<thead>
<tr>
<th>Commands</th>
<th>Responses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read*</td>
<td>0 or 1 arbitrary</td>
</tr>
<tr>
<td>Write 0</td>
<td></td>
</tr>
<tr>
<td>Write 1</td>
<td></td>
</tr>
<tr>
<td>Shift head left</td>
<td></td>
</tr>
<tr>
<td>Shift head right</td>
<td></td>
</tr>
</tbody>
</table>

Real-time equivalent:
(a) Fatter or faster heads.
(b) Larger alphabet. (Use some block code.)
(c) Multiple tracks. (Use cross-product alphabet.)
(d) One end marker. (For one simulation, use special "mark"; for other, "fold" tape.)

*(Could get by with yes/no queries)*
Note apparent advantages of the shared access:

1. Each head can read the other's work, without ever "copying" it.

2. Can always be ready to retrieve from start, even while continuing to store.

3. Can read data in various orders, without ever taking time to re-order it.

4.
Break for examples

Our "buffer"/"queue" (both heads one-way):

Recognize $L_1 = \{ x\#x' \mid x' \text{ prefix of } x \}$
(change once from "storage" to "retrieval").

$L_2 = \{ x\#x \}$ (maybe easier).

$L_3 = \{ \text{right-shuffle } (xy, \#x) \}$
(change once from stg. to equal-rate stg. + retrieval).

$L_4 = \{ x\#y\#z^R \mid y \text{ prefix of } x, \ z \text{ suffix of } x \}$
(more than just queue ops).

How well with 1-head tapes?
Some "real-time recognition" exercises
for a 2-head tape (machine):

\{x \# y \mid x = y\}
\{x \# y \mid x \text{ prefix of } y\}
\{x \# y \mid y \text{ prefix of } x\}
\{x \mid h_2(x) = h_u(x)\}^3, \text{ where }

h_2 \text{ maps lower-case letters to their parities (0/1) and erases upper-case letters, }

h_u \text{ does vice versa.}

\{x \mid h_e(x) \text{ prefix of } h_u(x)\}
\{\text{right-shuffle } (xy, \# x)\}

Note: All these require only a "buffer" (or "queue").

\[\{x \# y \# z^R \mid y \text{ prefix of } x, z \text{ suffix of } x\}^?\]

How well with 1-head tapes?
Storage structures

"storage structure": gives an appropriate 
"response" ($\in \Delta$) to each next "command" ($\in \Sigma$).

Simulation (by some finite-state program 
with its own storage structure):

"on-line": each command's response must 
precede next command

"in time T": at most $T(n)$ steps to 
handle $n$ commands

"real time": $O(1)$ steps between command 
and response (stronger than 
"linear time" ($\frac{T(n)}{n} = O(n)$)); 
technology-independent formalization 
of "without time loss")
\[ \exists x \neq y \mid x = y \]?

2 tapes!

All comparisons during \( y_2 \):

\[ x_1^R \text{ vs. } y_1^R, \]

\[ x_2 \text{ vs. } y_2. \]
\{x\#y \mid x \text{ prefix of } y\}?

Also 2 tapes!

Corollary of =.

\{x\#y \mid y \text{ prefix of } x\}?

Seems harder, but 3 tapes (!):

Phases that double cumulative no. of \textit{x} symbols read.

Phase invariant:

1. full 8-fold version
2. full 4-fold version
3. blank (or junk)

Phase:

- Leave 1 as “insurance”.
- Complete 2 to 8-fold version.
- Build 3 to 4-fold version.

\textit{Always “safe”!}
\{x \mid h_2^*(x) \text{ prefix of } h_u(x) \}\ ??

At least as hard as \( \{x \# y \mid y \text{ prefix of } x \}^3 \).

3 tapes still possible??

\{x \mid h_2^*(x) = h_u(x) \}\ ??

At least as hard as \( \{x \# y \mid x = y \}^3 \).

2 tapes still possible?? 3?

---

Other good candidates?

(Only "beg the question" with the "universal" (or "complete")
 candidate:)

\( \exists x \mid x \text{ is a command sequence for a } 2\text{-head tape,}
\text{ whose last query should be answered "yes" }^3 \).

Recognize this in real time

\( \Rightarrow \text{ can simulate } 2\text{-head tape in real time.} \)
Theorem. A multihead tape can be simulated without time loss using only single-head tapes.
Proof: Use $h-1$ 2-head tapes:

... #1  #2  #3 ...

Simulation of 2-head tape:
"Put the heads on separate tapes."
See Figure 1.
Obvious danger: Dirtiness $S$ might pass $\frac{1}{2}$.

Solution: Keep $S$ tiny:

Alternate between use of two such pairs.

Simultaneously, alternate "cleanup".

Invariant assertion for main loop ($S$ fixed, tiny):

1. One pair at most $S$-dirty. (Loop body will use this for current simulation.)
2. Other pair at most $2S$-dirty. (Loop body will "service" this.)

Appropriate constraints on loop body:

1. Short enough so that first pair gets no more than $2S$-dirty. This allows $t$ commands if $S\Delta + t \leq 2S(\Delta - t)$, or $t \leq E\Delta$.
2. Long enough to make second pair clean enough.
Distilled problem:

Clean up in time $O(\Delta)$.
(Any constant is fine; we can adjust the command frequency. *)

2-phase solution:

1. Fully update to call-time.
(Use SYNCH and a couple passes.)

2. Catch up to currency fast enough to keep dirt relatively small.
Again, time $O(\Delta)$ is fast enough.

*Unless $\varepsilon \Delta < 1$. While $\Delta$ is so small, use a “perfectionist” simulation.
How catch up in time $O(\Delta)$?

1. Save ignored commands ("backlog") on tape #5.

2. Run through them at full speed, using tape #6 to collect more backlog -- but at most half as much. Switch roles of backlog tapes, and repeat until convergence to no backlog.

$$\text{Time} = O(\Delta + \Delta/2 + \Delta/4 + \cdots) = O(\Delta).$$
So we have:

**Theorem 1:** Can simulate 2-head tape in real time, using just 4 single-head tapes.

New (but "obvious") **lower** bound:

**Theorem 2:** Cannot do so using just 2.

**Proof:** Much harder! (Typical.)

**Outline if time.**
What does it take to “fully simulate” a “2-head tape”? 

Not enough: 2 single-head tapes 

1. Wrong? 
2. Duh? 
3. Wow! 

Enough: 4 single-head tapes! 

What about 3?? 

(or a buffer? or 3 and a pushdown store? ...)
Favorite remaining questions:

Can 2 heads beat 3 tapes?
Or 1000 ... 1001?

Can 3 heads beat 3 tapes?
Or 1000 ... 1000?

Nonobvious languages to focus on?

Can 7 tapes simulate 3 heads?
Or 3995 ... 1000?