```
import concurrent.futures
from typing import List, Union
from swarms.structs.agent import Agent


def update_system_prompts(
    agents: List[Union[Agent, str]],
    prompt: str,
) -> List[Agent]:
    """
    Update system prompts for a list of agents concurrently.

    Args:
        agents: List of Agent objects or strings to update
        prompt: The prompt text to append to each agent's system prompt

    Returns:
        List of updated Agent objects
    """
    if not agents:
        return agents

    def update_agent_prompt(agent: Union[Agent, str]) -> Agent:
        # Convert string to Agent if needed
        if isinstance(agent, str):
            agent = Agent(
```

```
            agent_name=agent,
            system_prompt=prompt,  # Initialize with the provided prompt
        )
    else:
        # Preserve existing prompt and append new one
        existing_prompt = (
            agent.system_prompt if agent.system_prompt else ""
        )
        agent.system_prompt = existing_prompt + "\n" + prompt
    return agent

# Use ThreadPoolExecutor for concurrent execution
max_workers = min(len(agents), 4)  # Reasonable thread count
with concurrent.futures.ThreadPoolExecutor(
    max_workers=max_workers
) as executor:
    futures = []
    for agent in agents:
        future = executor.submit(update_agent_prompt, agent)
        futures.append(future)

    # Collect results as they complete
    updated_agents = []
    for future in concurrent.futures.as_completed(futures):
        updated_agents.append(future.result())
```

```
return updated_agents
```